IBReplicator 2.5

© 2007 IBPhoenix Editors

User Manual

Build 2.5.2
18/02/2007

B d 2.5.2 18/02/2007 © 2007 IBPh n Ed t rs

IBReplicator is a toolset for implementing and controlling database replication in Firebird versions 1.0x, 1.5.x and 2.x and Borland InterBase 5.x, 6.x and 7.x. It is software that integrates with your databases, allowing users to replicate data seamlessly and transparently between databases on both local and remote sites.

This manual provides an overview of IBReplicator and how IBReplicator and its tools can be used to meet your replication needs. It provides a step by step breakdown on how to install, configure and use IBReplicator. Included are descriptions of the various replication models you can implement and how to use the Replication Manager tool to set up your replication schemas and to operate and schedule replications.

Version notes for this release can be found at the end of this document.

Chapter 1 describes the platform, database engine, system and licensing requirements.

Chapter 2 is an overview of the IBReplicator toolset.

Chapter 3 describes the architecture and terminology of IBReplicator and Chapter 4 goes on to explain how the replication actually works

In Chapter 5, you will find a discussion of the planning you will need to consider in designing your custom IBReplicator system.

Chapter 6 contains instructions for installing the pieces of your replication system on source and target servers.

In Chapter 7, you will find the details for defining a replication schema&emdash;the rules.

Chapter 8 explains how to run and manage your replication server.

In Chapter 9 you will find a miscellany of topics concerning issues that will arise in more complex setups and how to deal with them.

Table of Contents

Chapter 1 - IBReplicator Requirements 6

Chapter 2 - Introducing IBReplicator 9

Chapter 3 - IBReplicator Architecture 12
- Terminology 12

Chapter 4 - How IBReplicator Works 16

Chapter 5 - Planning for Replication 20
- Choosing a Replication Scheme 20
- Database Design Issues 24
- Physical Distribution Considerations 26
- Conflict Resolution 26

Chapter 6 - Installation and Setup 29
- First Configuration Database 30
- Registering Databases 33
  - About Licences 39

Chapter 7 - Defining a Replication Schema 42
- Customising the Default Settings 43
- Creating the Schema 45
- Choosing the Source Database 50
- Choosing the Target Database 54
- Choosing Replicated Tables 58
- Table Settings 62
- One-time Synchronization 65
- Identifying Primary Keys 69
- Choosing Replicated Columns 71
- Creating Source System Objects 72

Chapter 8 - Managing Replication 75
- Windows Operation 75
  - Command-Line Operation 79
- Linux Operation 81
- Management Tools 84
  - Replication Monitor 85
  - Replication Scheduler 86

© 2007 IBPh en Ed rs

Notify Server 88
Conflict Resolution 89
License Manager 91
Schema View 94

Chapter 9 - Advanced Topics 99
Schema Numbering 99
Metadata Changes 102
Adding Timefields 102
Advanced Mapping Techniques 104
Complex Schemas 106

Version Notes 112

Index 114

© 2007 IBPh en Ed rs

1

Chapter 1 - IBReplicator Requirements

Chapter 1 - IBReplicator Requirements

Operating System

IBReplicator Server runs on

- Microsoft Windows 95, 98, NT, 2000, XP, Server 2003
- Linux

The current version of IBReplicator's Replication Manager needs to run on a Windows host. It can access a remote IBReplicator server on Linux.

Database Engine

IBReplicator supports replication to and from databases running under

- Firebird 1.0, 1.5, 2.0
- InterBase 5.x, 6.x, 7.x

Disk Space

To install the IBReplicator software you need at least 2MB of available disk space.

Database Installation

Both the Replication Server and the Manager require at least the appropriate Firebird or InterBase client library to be installed. Installation and configuration will involve creating a configuration database, so these activities must take place on a running Firebird or InterBase server.

Licensing

There are two types of licence: Replicator and Replicant. Licences must be installed into the configuration database that you want used for your active replication.

¬ A Replicator licence allows IBReplicator to run

¬ A Replicant licence allows a database to be accessed as a source or target

As a rule of thumb, you require one Replicator licence to run a replication server and one Replicant licence for each database that is a source or a target. Each Replicator licence comes with one implicit Replicant licence that will be included in the count when the IBReplicator checks that there are enough Replicant licences available.

Thus, the minimum IBReplicator configuration of one source database being replicated one-way to one target database requires a single Replicator licence and one more Replicant licence.

A two way replication system—where, for example, Database A is being replicated to a target Database B and Database B is being replicated back to Database A—still requires

© 2007 IBPh en Ed rs

one Replicator licence and a count of two Replicant licences.

Many other configurations are possible. Additional Replicant licences are available. Checking is concerned only with ensuring that the count of Replicant licences is at least equal to the number of databases configured in the schema.

Chapter 5 20 describes some possible configuration models, which may help to clarify the licensing requirements.

There are more notes about licences in Chapter 6 39, About Licences. Instructions for installing them can be found in the topic about the License Manager 91 tool.

© 2007 IBPh en Ed rs

2

Chapter 2 - Introducing IBReplicator

Chapter 2 - Introducing IBReplicator

IBReplicator is a management system that is is added to servers hosting Firebird or Interbase databases. It keeps track of data changes in databases designated as sources and enables users to synchronise two or more databases at one or more sites, locally or remotely.

Replication makes it possible to maintain identical copies of databases, database tables or even pre-selected sets between source databases and target databases. The Replication Server's typical activity is to work behind the scenes, replicating data at set intervals, or according to a schedule, or in response to specific events.

IBReplicator is designed as a software layer, transparent to ordinary users, that sits between databases and applications. Data integrity is managed by using the two phase-commit protocol that is available to Firebird and InterBase. The effect of two-phase commit is that data for a replication remains uncommitted on the target until the replication transaction has been committed on the source database. This ensures data integrity should there be a problem in the source or target database or a network failure.

An important component of the Replication Server suite is a graphical, Windows-based Replication Manager program for defining, managing and monitoring the operations and for resolving replication issues.

IBReplicator connects to Firebird or InterBase directly through the API. No middleware or third party driver is used, overhead is low and the system is small, robust and fast.

Support for Database Features

IBReplicator's replication can handle

- any data type supported by Firebird or InterBase, including BLOBs and arrays
- columns defined using quote-delimited reserved words as identifiers
- other SQL quote-delimited identifiers
- multi-segment primary keys
- international character sets

Replication can be configured to respond to database events.

Replication to or from databases other than Firebird or InterBase is not supported currently. However, the replication engine was largely re-written for IBReplicator 2.0 and replication to other databases (MySQL, Microsoft SQL Server, Oracle, et al.) could be implemented. The IBPhoenix development team would like to hear from you if you have a specific requirement for this functionality.

IBReplicator Security

Security provisions use the same mechanisms as those used to secure your normal

© 2007 IBPh en Ed rs

Firebird and InterBase databases. User names and passwords that IBReplicator uses to connect to source and target databases are stored in the respective Firebird or InterBase security databases.

Benefits of IBReplicator

Replication can enhance the usability of your system in a number of ways. For example,

Data availability - Work groups can have their own copy of a replicated database, enabling them to function independently within a larger network. A local copy of a database means that the work group does not have to compete for larger network resources or suffer response delays by accessing databases on remote sites. Decision support applications can be separated from high-volume transactional applications if necessary, which can reduce network traffic and improve data availability and system performance.

Reliability and failover - Replicated databases can be maintained for improved system fault tolerance. Applications can switch from original source databases to replicated copies in the event of system failure.

Flexibility of data distribution - Any organisation has its own replication requirements and could be operating over a variety of databases. With IBReplicator you can replicate full databases, a subset of tables, selected rows and columns from tables in a source database to a number of target databases. You can specify when you want automatic replication to occur; on a predefined per second schedule (virtually continuously), on a time-based schedule (hourly, daily, weekly), or when a specific event occurs; or you can specify to initiate replication manually. You can define multiple replication schemas, if you need to, that will enable different source database subsets to be replicated to specific multiple targets.

© 2007 IBPh en Ed rs

3

Chapter 3 - IBReplicator Architecture

Chapter 3 - IBReplicator Architecture

This chapter introduces the components of IBReplicator and explains the role of each in the replication system.

IBReplicator Server

The IBReplicator Server (ReplServer) is the program that performs replication operations. To determine what schemas need to be replicated where, it uses a configuration database. A single server can replicate multiple source databases cyclically. Available on windows as a service (NT, XP, 2000, 2003) or as an application, or on Linux as a shell window or a daemon.

IBReplicator Replication Manager

The Replication Manager is a Windows graphical tool for the user to define, manage and monitor replication schemas. It also includes a scheduler for defining custom replication schedules by the clock, rather than specifying for a replication to run every n seconds.

Configuration Database

IBReplicator's configuration database is where the program saves and maintains the details of what is to be replicated and where it should be replicated from--the source --and to--the target. The specifications for a replication operation are referred to as a replication schema.

You can have IBReplicator create multiple configuration databases to handle multiple schemas, or you can store multiple schemas in one single configuration database.

IBReplicator Processing Tables

IMPORTANT: For IBReplicator to work properly, each base table that is to be involved in replication must have its own unique or primary key.

The Create System Objects function generates three triggers--an insert, an update and a delete trigger--to track changes that are made to the database tables. Changes are recorded in the Log Table.

The Log Table stores the key information on all the changes that have been made to the tables in the source database.

The Manual Conflict Resolution Log contains a list of operations from the Log Table that were not successfully replicated because of a data conflict problem and now require manual user intervention to correct the problem.

3.1 Terminology

Source (database)

When we talk about the source database, we refer to the database containing the "master

© 2007 IBPh en Ed rs

version" of data that is intended to be copied to one or more other databases.

Target (database)

A target database is one that that receives newer versions of its tables from a source database, generally on a regular basis.

Replication

Replication is the process of passing newer versions of tables from a source to a target database, usually by updating records that have changed in the target tables, inserting records that exist in the source but not in the target and deleting records that exist in the target but not in the source.

Synchronization

Synchronization is an occasional (usually once-only) method of replication, where the entire contents of tables from a source database are pumped into "empty" tables in the target. Synchronization is useful for adding data to a new target database from a source database to ensure that the two databases are in synch. If a target database has simply been offline, there is no need to perform synchronization, as normal replication will catch up with any backlog of changes.

Replicator

Replicator is a term for a database that is configured in IBReplicator to be a source database.

Replicant

Replicant is a term for a database that is configured in IBReplicator to be a target database.

Registering a database

is the action of configuring an IBReplicator installation to find and recognise a database.

Asynchronous Replication

For some systems it's fine for replication to happen every so often, perhaps daily, weekly or monthly or only on request. This asynchronous replication requires only a low-end network setup, even just dial-up. Replication can be scheduled to take place during quiet periods, at night or at a weekend, for example.

Of course, the lag between replication occurrences means that the target databases are likely to be out-of-date most of the time.

Synchronous Replication

In situations where databases must always be kept up-to-date, synchronous replication will be needed, to replicate changes as and when they occur. This can put pressure on the network, since this constant replication can generate a lot of traffic.

© 2007 IBPh en Ed rs

Generally a compromise solution can be reached, where critical data is replicated synchronously and less critical data is replicated asynchronously. For such replications, the configuration needs to be stored in the source database.

Data Subset(ting)

When only certain rows in a table are replicated to a target, it is called data subsetting. It is entirely possible that replication from a single source table could replicate different subsets to multiple targets.

© 2007 IBPh en Ed rs

4

Chapter 4 - How IBReplicator Works

Chapter 4 - How IBReplicator Works

The purpose of IBReplicator is to create and maintain identical data in database tables distributed across multiple local and remote sites. This chapter provides a technical overview of how the various replication components work together to allow replication to take place.

Replication Schema Definition

A replication schema defines

- a source and target pair of databases
- the objects in the source database that are to provide data for replication
- the objects in the target database that are to receive data from each particular source object
- the rules for running the replications

As a replication schema is defined, the Replication Manager stores its details in a configuration database. Once a schema's outline is defined, you tell the Replication Manager to create system objects in the specified source database[s] and the schema is completed.

System Objects

System Objects can be regarded as the metadata of a replication schema. They comprise triggers and tables that ensure that, for each changed row in the source database, the table identifier, primary key and the action (insert, update, delete) are logged.

The logic of the triggers handles new, updated or deleted data created on the local (source) database and, if applicable, incoming data from other databases that are replicating back to the local database. The log table forms a queue of committed work waiting to be replicated.

As an insert, update or delete takes place on a specified table or set, a trigger fires to write information to a log table when the transaction is committed. It is this underlying mechanism that prevents information from being replicated before it is committed in the source database: no commit, no log entry, no replication.

Executing Replication

The replication server can receive the instruction to replicate from several alternative sources:

- Its own internal timer
- An explicit request to replicate
- A timed signal from the scheduler

© 2007 IBPh en Ed rs

- In response to an event

Sequence of events

Whenever the server receives the instruction to replicate, it follows this sequence:

1. It looks at the configuration database to determine the source and target pairs of databases.

2. Next, it looks at the replication log table in the source database to find out which changes in the source database need to be replicated to the target.

3. The server processes the rows by querying their data from the source database tables and duplicating the logged action (insert, update, delete) to the target database, according to the specifications defined by the replication schema.

   Rows are processed in FIFO order, i.e., in exactly the same order that they were put into the log by the triggers. The operation specified by the log record is applied to rows and columns of the tables in the target database :

   - An insert is duplicated by inserting a new target row having the same primary key values and the same values for each column that is specified for replication.

   - An update is replicated by finding the row with the same primary key and updating the replicated columns to their new values.

   - A delete is replicated by finding the row with the same primary key and deleting it.

Clearly, it is essential that both source and target table must have primary keys and that those keys be compatible. Replication cannot be performed between unkeyed tables.

Two-phase Commit

To ensure the integrity of the distributed replicated data and the consistency of database state on both sides, the replicator makes use of the two-phase commit feature of Firebird and InterBase transactions. Two-phase commit is analogous to a contract between both (or all) of the databases involved in a distributed transaction. Each database commits its part of the transaction and waits for the other (or others) to signal that they have committed their part(s), before each proceeds to secure its own commit.

The replication is complete when the whole log has been processed and both (or all) sides of the transaction have completed the commit and verification phases of the commit.

The server deletes the rows from the the replication log queue when it has finished with them. The whole log can be kept intact throughout the replication or a commit interval can be specified to determine the number of rows to process before rows are deleted from the queue. Because of the two-phase commit, any problem that arises while the queue is being processed causes an error. That, in turn, prompts a rollback. Hence, the queue is always retrievable for a re-run, with the secure knowledge that the replication has either run through to a successful completion and emptied the queue or has failed

© 2007 IBPh en Ed rs

as a whole and retained the queue.

Limbo Transactions

If a failure occurs during the two phase commit, it is possible that you may get left with a limbo transaction. That means that the pieces of work committed or rolled back by the first phase of the commit are left in each database in an unresolved state. Although the database engine can tell whether the intention was to commit the transaction or to roll it back, it does not presume to complete the second phase on its own initiative.

To correct a limbo transaction you need to use the automated two phase recovery mechanism provided by the -t[wo-phase] switch of the gfix command line utility that is installed in the bin directory of a full Firebird or InterBase server installation.

To have gfix try to resolve limbo transactions in the databases, go to the bin directory of the respective database servers involved and use the command as follows:

Windows

```
gfix -t -all database_filespec
```

Linux

```
./gfix -t -all database_filespec
```

The order in which you attend to each database is not important.

> There are other options for recovering or resolving limbo transactions. Consult The Firebird Book (Chapter 39) or the InterBase Operations Guide (OpGuide.pdf) for details.

© 2007 IBPh en Ed rs

Chapter 5 - Planning for Replication

Chapter 5 - Planning for Replication

This chapter describes the planning and preparations that are necessary for implementing replication schemas that IBReplicator can manage and process.

Choosing a Replication Scheme 20

Database Design Issues 24

Physical Distribution Considerations 26

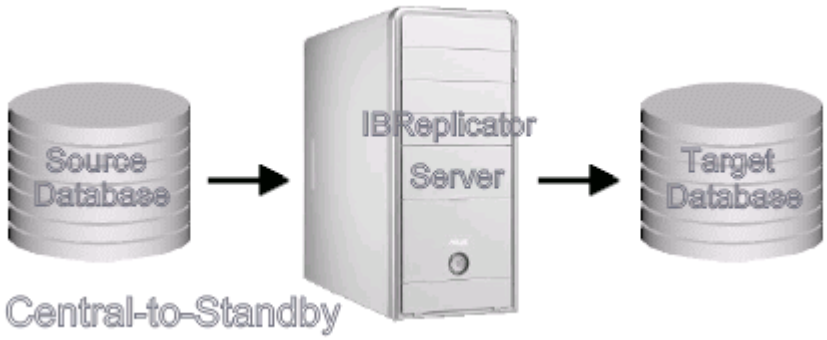
Conflict Resolution 26

5.1 Choosing a Replication Scheme

You can choose from a variety of replication models to propagate data throughout your network. Your choice will reflect the requirements of your system. Models can be combined to fulfil needs that are too complex to be satisfied by a single scheme. Multiple schemas can be set up to fulfil different requirements, too.

Central-to-Standby

This simple model fulfils a requirement to replicate one way from a source database to a target. It is the scheme typically used to fulfil a "failover" requirement, whereby a standby database must be available to resume activity if a source database should fail.

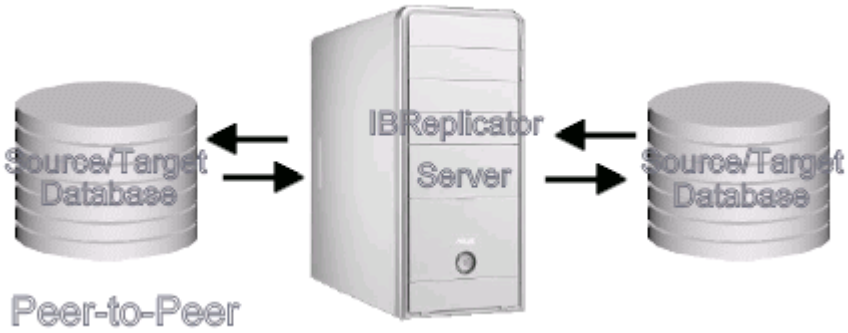


A Central-to-Standby scheme can be set up with a Peer-to-Peer schema (see below) to enable changes in a failover database to be applied to the source database once it comes back on line.

Peer to Peer

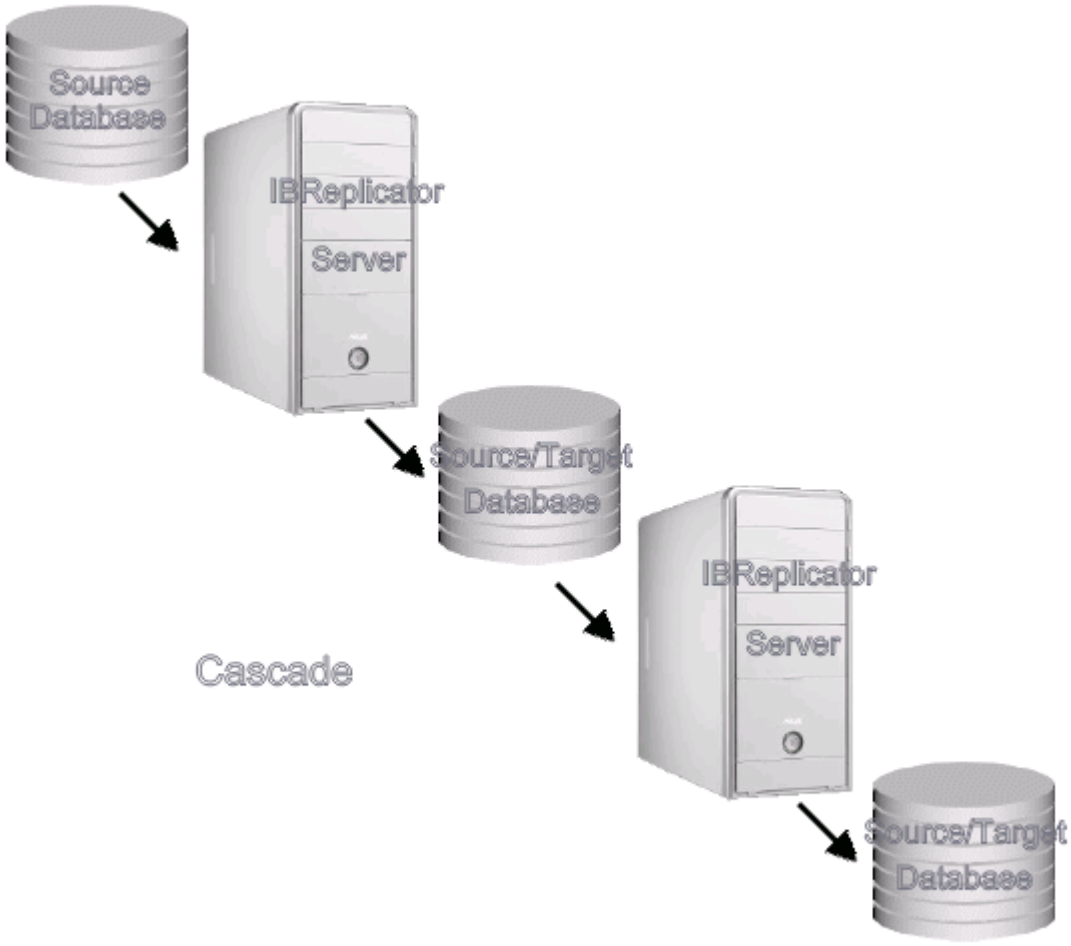
In a Peer-to-Peer environment, bi-directional (two-way) replication allows source databases to send changes to targets and vice-versa. Each database (source or target) can function autonomously at its own site. Changes from one database are replicated to the other to bring all databases up-to-date.

© 2007 IBPh en Ed rs



Cascade

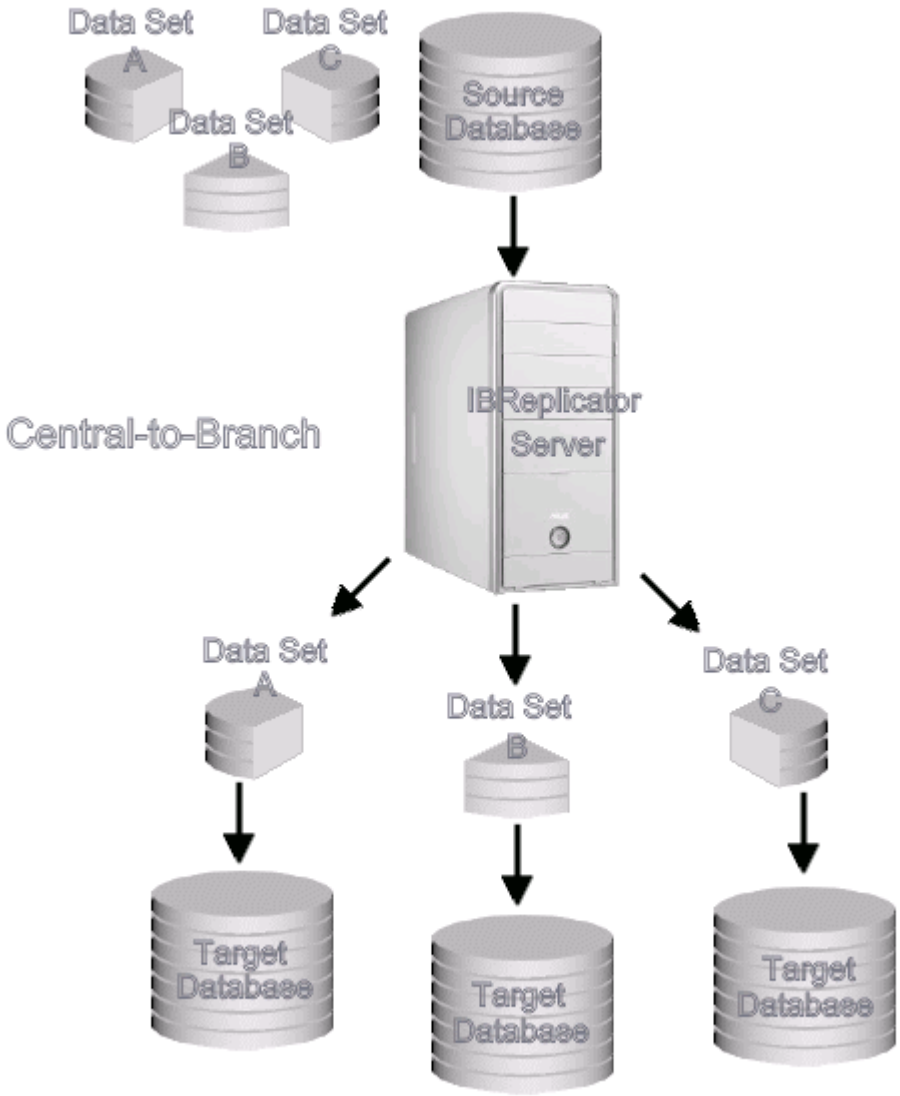
Cascade replication takes data from a source database and moves it to a target that, in turn, becomes a source for another target, and so on, down a hierarchy. An example of cascading replication might be where Head Office replicates to a regional office which then replicate to a branch. The cascade model might be combined with a Central-to-Branch schema (see below) so that each level is replicating separate sets of data to a number of levels below it.



Central-to-Branch

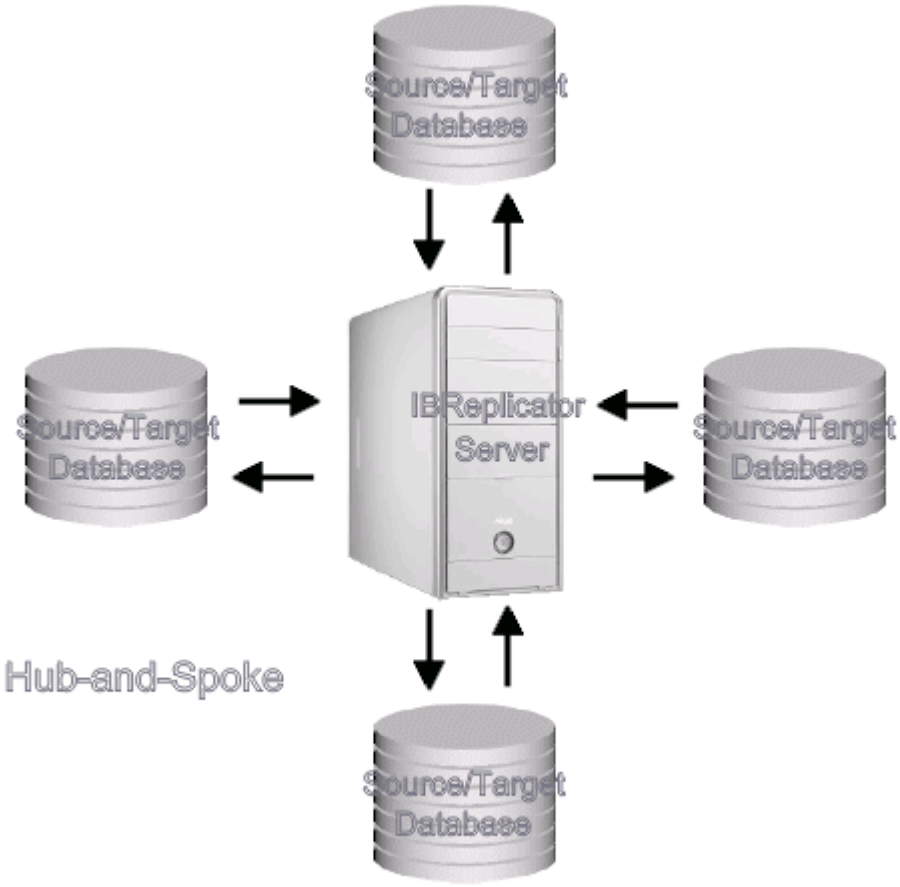
In a Central-to-Branch schema, subsets of the central database are designated for replication to branch sites. Data at the centre is horizontally partitioned, branch by branch. Usually, centrral data sent to branches is read-only. However, this schema is sometimes combined with Peer-to-Peer, so that branches update their sets locally and it is replicated back to the centre.

© 2007 IBPh en Ed rs



Hub-and-Spoke

In a Hub-and-Spoke schema, a central database (the hub) replicates bi-directionally to multiple databases. It has a Peer-to-Peer relationship with each of its spokes. Cascade replication is implicit in this environment, because each of the spokes receives replicated data whenever the hub database or any of the other spokes is updated.



5.2 Database Design Issues

Certain aspects of your database design and structure will be important as you prepare for replication--particularly the unique or primary key constraints that will logically link source and target sets and the TIMESTAMP columns that you may wish to use for conflict resolution.

Unique Keys

All tables must have unique keys. As a rule, the unique key used to map a source row to a target row should be the primary key.

> IMPORTANT: Firebird allows null elements in uniquely constrained keys. Such keys cannot be used for mapping source and target tables for replication.

Distributed Primary Keys

Maintaining keys that are unique across database boundaries are always going to be the major issue when you are setting out to merge records from different databases. In

© 2007 IBPh en Ed rs

scenarios where rows are being inserted in multiple databases using generated primary keys, it will be necessary to implement some changes to avoid unresolvable key violation conflicts.

- In a Central-to-Branch scenario a multi-segment primary key can be implemented, consisting of a Branch code + a generated ID. For existing databases, this solution will entail a large amount of work, both in the databases and in existing applications.

- An alternative to augmenting the primary keys is to use GUIDs. Implementing a GUID primary key would avoid impacting the existing single-column PK structure and possibly reduce the amount of change required in application code. On the database side, it requires a type change--simpler to do if you are using domains-- and alteration of any trigger and stored procedure code that you have to generate values for keys.

- It is possible to use a uniquely constrained key that is not the primary key. This might be necessary in some heterogeneous schemas.

- Another alternative is to replicate changes, from all of the targets that are going to receive an update replication, back to the central database first, before anything gets replicated back out to the targets. All inserts from the targets go into the central database via an insert trigger which generates a new primary key for them from the central database. Replication out to the targets just has to be able to ascertain which of the "new inserts" originated from the target in focus and treat them as updates.

  The main need would be to tag records on both sides, so that both the central and the target databases know that they need special treatment. How (or whether) this could be done would depend on considerations such as the impact if records that had been locally inserted by the target changed their primary keys after replication, whether a target's original key needed to persist for local usage, and so on.

Timestamps for Replication Use

If your chosen method of conflict resolution 26 is going to be by Timefield, it will be necessary to include a TIMESTAMP column in every replicated table—both source and target—and to write triggers to ensure that every INSERT and UPDATE operation on those tables gets timestamped.

> IMPORTANT The identifier (name) of both columns in the source-target pair must be identical.

DATE vs TIMESTAMP

Timefield conflict resolution does not fail if either or both of the compared fields are of DATE type rather than TIMESTAMP. However, use of the date-only type is not recommended because replication might not occur where the source and target rows have the same date in the timefield.

The choice is not an issue if the databases are all Dialect 1, since a date-only type is not supported there.

© 2007 IBPh en Ed rs

Clock Synchronisation

Don't overlook the need to implement mechanisms to synchronise the system clocks of the source and target computers and, where necessary, to adjust for time zone differences.

5.3 Physical Distribution Considerations

The general purpose of replication is to ensure that changes made to a source database are duplicated to another database, so that the target database contains the same data as the source. Although the source and target databases can be on the same system, they are usually spread around a local network or distributed across a wider network.

Distribution Conditions

Replication can be organised to distribute data to meet distribution needs in a virtually infinite variety of ways.

One-way Replication

In one-way replication, one database is always the source. In physical terms, its targets receive data that they do not overwrite, append to or delete. Replication refreshes the content of the targets without further modifying the source database.

N-way Replication

An n-way replication scheme involves databases that act as both a source and a target to other databases. One form of n-way replication is where two databases replicate back and forth to each other. The "n" can be more than 2, since reciprocal replication schemes often involve schemas where the same databases can be sources and targets to a number of others.

Heterogenous Replication

Source and target databases do not always have the same structure. One may contain tables that don't exist in the other. Tables may have different numbers of columns and different names for columns. A source dataset may replicate different sets of data to different targets. Any of these heterogeneous schemas is possible.

5.4 Conflict Resolution

Sometimes, a replication server may find a problem in trying to replicate an update to a row that is either missing or has already been updated. This is likely to show up in situations where the users of a target database are performing inserts, deletes and updates without being aware that changes may be happening elsewhere in a source database.

© 2007 IBPh en Ed rs

Issues like this generate conflicts during replication. Not all conflicts will be of the same type and different types of conflicts can be handled in various ways. Each source and target pair of databases can have its own conflict resolution settings.

Handling Conflicts

IBReplicator provides three mechanisms for handling potential conflicts.

Priority-based Resolution

Databases can be given priorities. The database with the higher priority number takes precedence. For example, if a source database has precedence, conflicts are resolved as follows:

- An update finding no identical key record in the target database is converted into an insert
- An insert finding a record in the target database with an identical key is converted into an update
- A delete finding no identical key record in the target database is ignored.

NOTE In versions 1.4 and below, the significance of the priority number was the reverse of that in later versions: the lowest priority number had the highest priority.

Master-Slave

The source database always takes precedence: resolution will be as described in the example above.

Timestamping

This method determines the newer version of data by comparing server timestamps written into the data. It needs a suitable timestamp column defined in both source and target tables. Older rows in the source database will not overwrite newer rows in the target.

© 2007 IBPh en Ed rs

6

Chapter 6 - Installation and Setup

Chapter 6 - Installation and Setup

IMPORTANT!

The configuration database structure in version 2 releases of this software is not compatible with that of version 1.x releases.

Please take copies of your configuration databases before installing.

The newer Replication Manager will automatically upgrade old configuration databases on opening.

Windows Installer Program

The IBReplicator software for Windows comes as an executable Windows installer program compressed in a WinZip file named IBReplicationServer-n.n.n-Win32.zip, where the 'n' characters represent numbers. The first 'n' represents the major release number, e.g. 2; the second the sub-release number; and the third, if present, identifies it as a patch release.

Inside the zip file is the Windows installer program, named as IBReplicationServer-n.n-Win32.exe. The installers do not normally contain the patch number in their names.

Decompress the zipfile into a temporary location of your choice, using your favourite decompression tool--WinZip, 7Zip, WinRAR, etc.

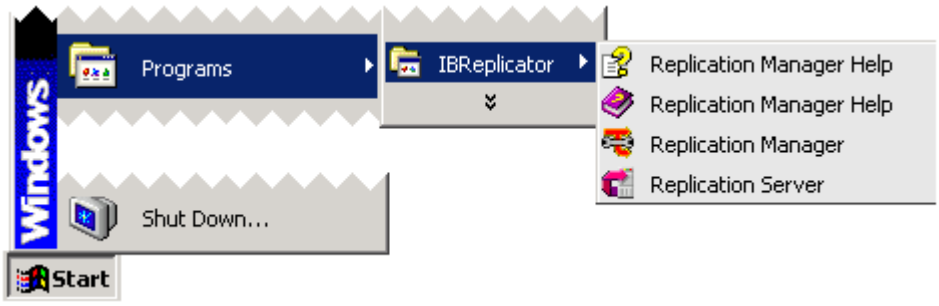
To install IBReplicator to your hard disk, double-click the IBReplicationServer-n.n-Win32.exe file's icon in its directory

Follow the dialogs as the installer runs, providing a custom root location for the software if you don't want the defaults.

Once the installer has finished, close it. If instructed to do so, close down any applications you have running and reboot the machine.

Start Menu Shortcuts

The installer program should have installed these shortcuts for you:



As a quick summary, here is what each of the shortcuts takes you to:

This help file in Winhelp format

This help file in compiled HTML (.chm) format

Starts the IBReplication Manager console program. See the next topic, First Configuration Database 43.

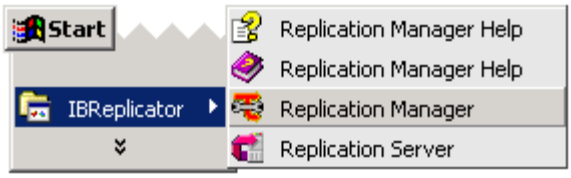
Starts the IBReplicator Server as an application. Used when running a replication that is not the default one.

6.1 First Configuration Database

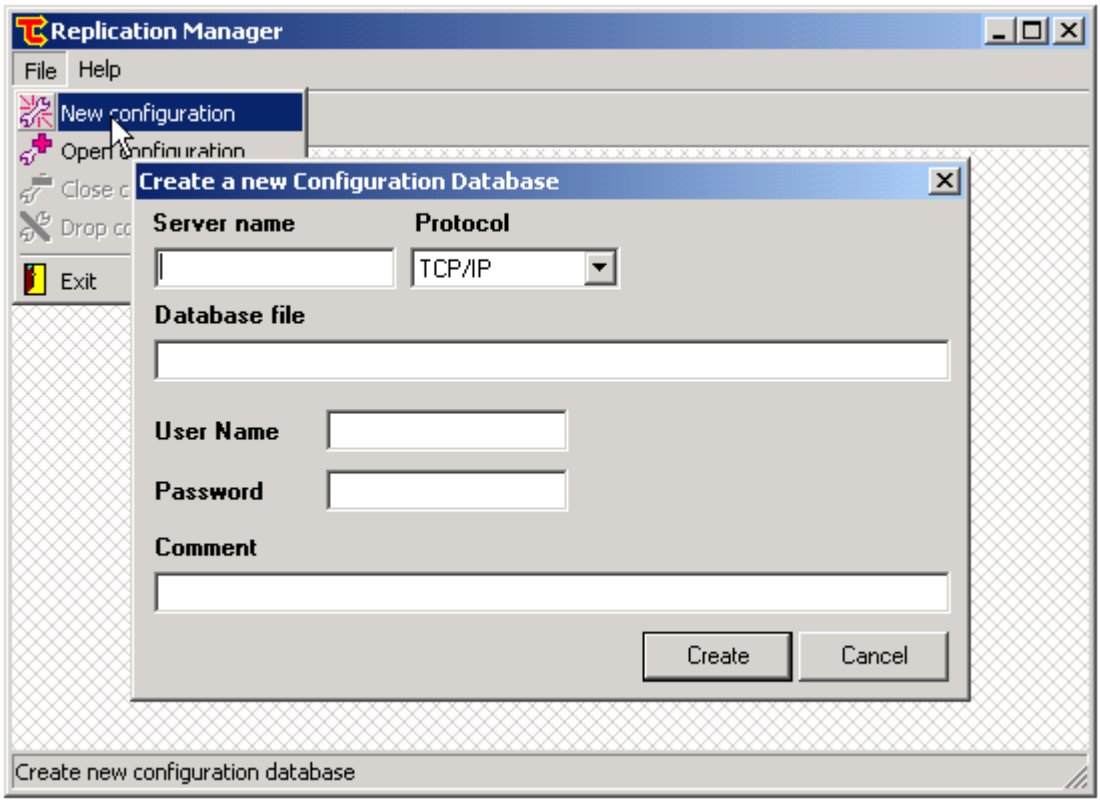
The Replication Server will use the configuration database to determine what it is supposed to replicate, where it should find the data (the source) and where the data is to be replicated to (the target).

Creating a Configuration Database from Scratch

To create a configuration database, start Replication Manager from the Start Menu:

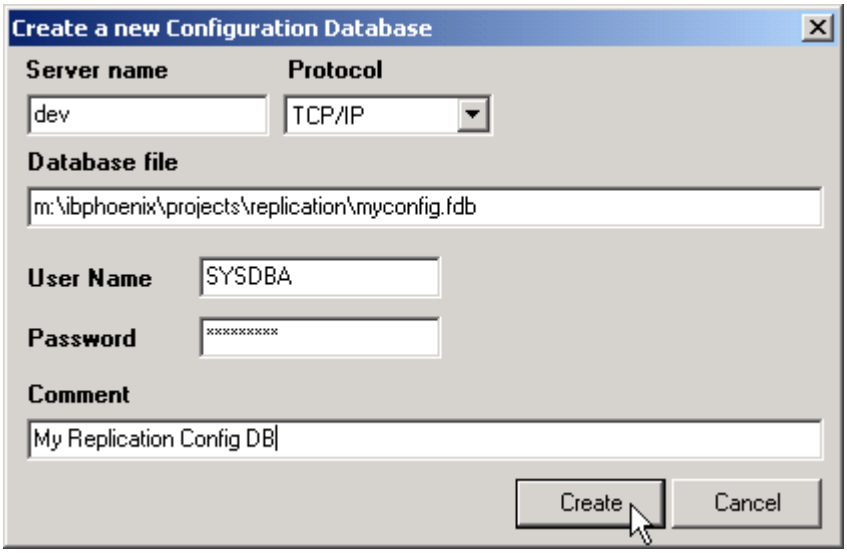


When the console appears, use File|New Configuration:



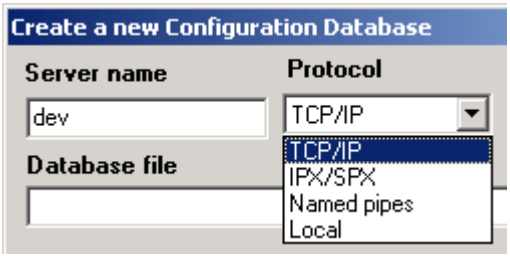
The setup dialog for creating the configuration database prompts you to define the server name you are connecting to, the connection protocol you intend to use, a file path specification for the configuration database and a valid username and password.

For example, suppose you decide to store your configuration databases in a subdirectory "replcfg" beneath some other directories to do with IBReplicator. Your setup entries might be similar to this:



Don't include the server name in "Database file" field: separate the two using the fields provided.

The one thing you might have to make a decision about is the Protocol. If you need to use a network protocol other than the default TCP/IP, drop the list down and select the one you need:



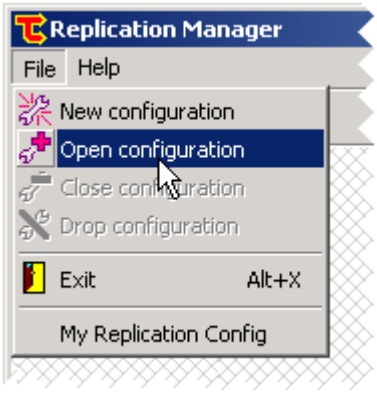
Note that "Named Pipes" is usually called "NetBEUI" in Borland circles, although NetBEUI isn't actually a protocol. "IPX/SPX" is a protocol for older InterBase versions that run under Novell Netware 4 or lower. If you are uncertain about which protocol you should choose, check with your network administrator or advisor.

Creating a configuration database adds it to the list of known configuration databases, any of which can be accessed quickly and simply through the File menu. The menu also provides for opening and closing configuration databases and for dropping (deleting) them.

When you are ready to create your configuration database, just click the "Create" button and you're done.

Using an Existing Database

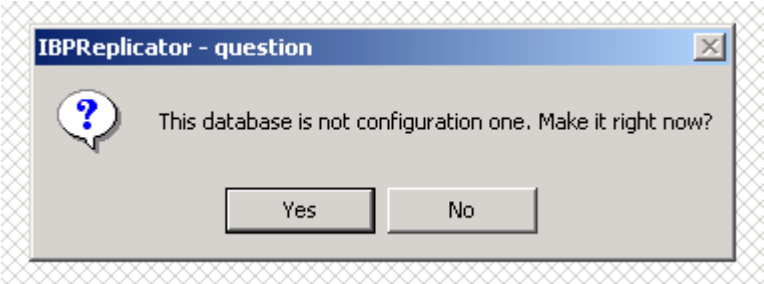
It is possible to store your replication configuration in an existing database, specifically the source database for a synchronous replication [13]. For this, use the Open configuration dialog from the File menu:



You will get a dialog almost exactly like the one for creating the regular configuration database. Fill in the fields to access the source database, remembering to include a valid username and password and submit it. You might like to include a useful nickname for the configuration in the Comment field, e.g. "Synch Checkouts" might be appropriate for a replication that is to be used to keep point-of-sale terminals in synch with a Sales database.

You will be asked if you want to make it into a configuration database:

© 2007 IBPh en Ed rs



Click Yes to have the program create the replication metadata in the source database. If you don't want to proceed, click No to abandon the idea.

NOTE Normally, you won't want the overhead of performing all replication synchronously; it is perfectly OK to break your replications out so that most of the replication runs asynchronously and the source database configures just the essential real-time replication.

The database will be added to the selection list beneath the File menu.

6.2 Registering Databases

The registration of a database identifies the database to IBReplicator and provides all the information needed to connect to the relevant databases. All databases that are involved in replication need to be registered, both source databases and target databases.

Once the configuration database has been created, you can register the databases that are going to be involved in replication.

To start registering a database, stay on the Databases tab. Above the right-hand pane is a toolbar.

The Toolbar

The toolbar gives access to the commands available during database registration. They provide shortcuts to the same commands that are available from the Databases menu. The active buttons at any point appear in an "enabled" state in monochrome. Hovering the mouse over an active button causes it to appear in colours.

Add Database is for adding a new database to the configuration

Edit Database is for doing any modifications required to the database currently selected

Remove Database is for taking a database out of the configuration. It

doesn't drop the database, of course!

Save posts and commits the current database registration

Cancels all current pending edits or removes the current unposted new database

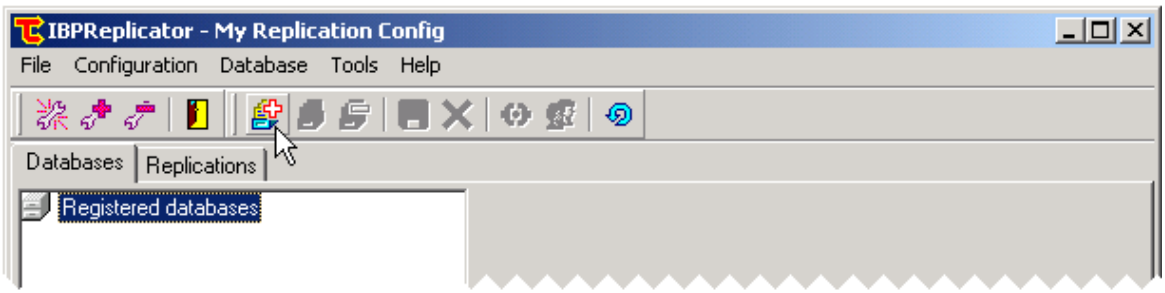
Test Connection does the same thing as right-clicking on a database and selecting Test Connection

Returns a count of connected users

Refreshes the list of registered databases

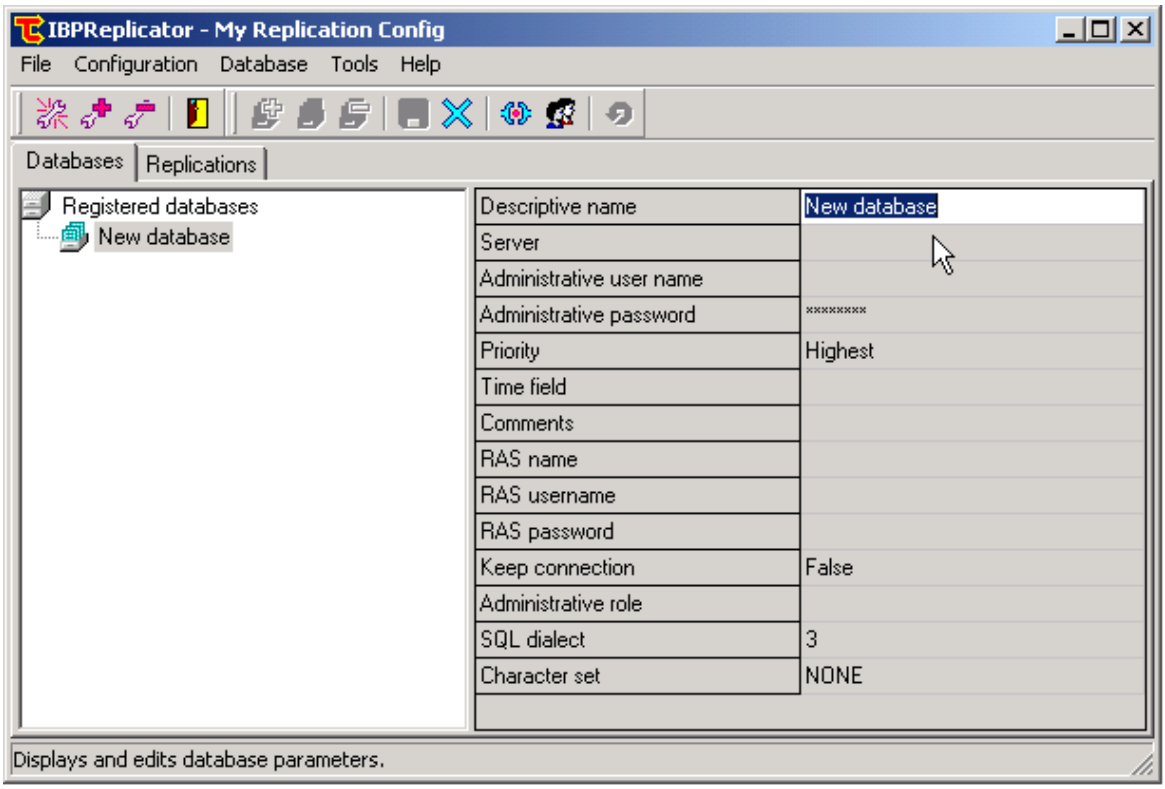
Registering a New Database

Click the Add button in the toolbar:



A new database record is initiated in the fields editor, ready for adding the necessary information for the replication server to find the database:

© 2007 IBPh en Ed rs



Fields for the Database Record

Descriptive name (required): A "user-friendly" name (not the database file name) that will be meaningful to anyone managing the replications. For example, "Head Office Source" is more meaningful than "New database"! It is not a formal identifier, so you may include spaces and punctuation as you wish.

Server (required): This is the network path and full file path of the database. It follows the syntax used by Firebird and InterBase to connect to databases and needs to accord correctly with the format required by the defined protocol and the file system of the host server that is running the database engine.

Suppose, for example, that the database file name is "source.fdb" and it is located on a Windows server named "dev" in a directory called G:\IBPhoenix\ibreplicator\data.

For TCP/IP protocol, the server entry should be

```
dev:G:\IBPhoenix\ibreplicator\data\source.fdb
```

For Named Pipes protocol, the server entry should be

```
\\dev\G:\IBPhoenix\ibreplicator\data\source.fdb
```

For an older InterBase database, named "source.gdb" on Novell, with the G-drive mapped as "vol5" it should be

```
dev@vol5:\IBPhoenix\ibreplicator\data\source.gdb
```

For a Linux or UNIX host, our example path would be of the form

```
dev:/IBPhoenix/ibreplicator/data/source.fdb
```

Administrative user name (required): The database server authentication username that the Replication Manager should use when connecting to and administering the database being registered. It can be a different user name from the Replication username 106.

Administrative password (required): The password for the administrative user name. The username and password belong together, as the two parts of a user login defined in the security database of Firebird or InterBase. See also Administrative Role (below).

The following two fields—Priority and TimeField—are required according to the conflict resolution 26 specified:

Priority: Used when priority-based conflict resolution is being set for the replication. In this case, replication conflicts are resolved by ensuring that data from the database with the higher priority number is preserved when conflicts arise.

Timefield: Used when time-stamped conflict resolution is being set for the replication. Your tables should all include a column with this name that contains the date and time when each row was inserted or last updated. If this functionality has been implemented, IBReplicator will ensure that the more recent row is preserved when conflicts arise. Refer to the design notes about Timefields 25 for discussion about choosing the right date/time type for this style of resolution.

NOTE In versions of IBReplicator prior to v.2.5 you can put any identifier in this field, as these older versions ignore it. For future-proofing, it is recommended that you use consistent Timefield identifiers across all tables in the source and target databases and enter the correct identifier here.

Comments (not required): This property is entirely for your convenience, to document and explain each database you register.

Administrative role: Similarly to the administrative user name and password, this is the SQL role that is used by the Replication Manager tool when administering the database being registered. The replication server uses another "replication role" that is provided when databases are specified in a replication schema.

SQL dialect (required): Informs IBReplicator whether it can use the functionality that is differently implemented in Firebird and in InterBase versions 6 and higher, such as the TIMESTAMP date/TIME format, restriction of double quotes usage, etc., (dialect 3) or whether it should confine itself to InterBase 5.x functionality (dialect 1).

Note that some Dialect 3-capable servers may be running Dialect 1 databases. Setting the replication dialect to 3 for such databases will cause exceptions.

© 2007 IBPh en Ed rs

Character set (required if the default character set of the database is not NONE):
The character set that is defined as the DEFAULT CHARACTER SET for the database.

Remote Access Parameters

IBReplicator can use the Windows Remote Access Services (RAS) to replicate across a dial-up connection. The following database parameters control how it works:

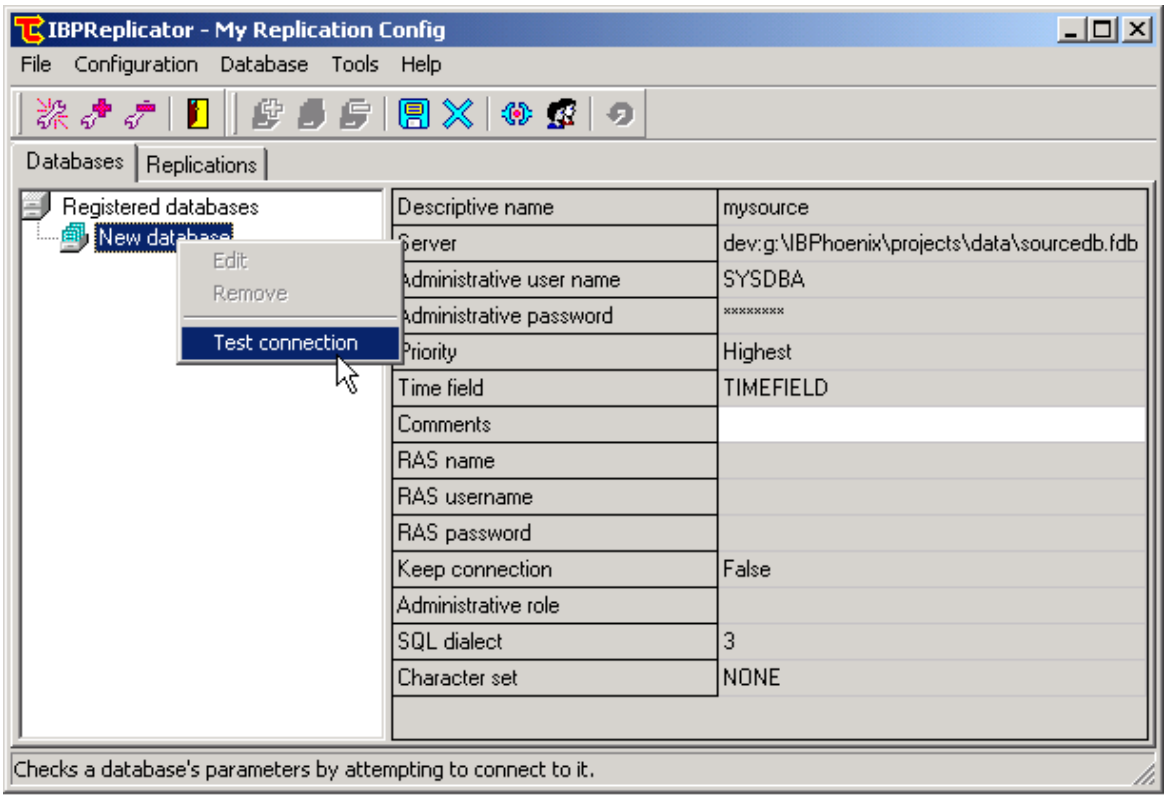
RAS name	This is the unique name of the dial-up connection as defined in the Windows "Dial-up Networking" dialog
RAS username	The dial-up username that the replication should use when dialling out
RAS password	The password that the replication should use when dialling out
Keep connection	Controls whether the replication hangs up as soon as it is disconnected from the database, or stays connected until the end of the replication cycle (all schemas) leaving a user to terminate the RAS connection

Checking the Connection

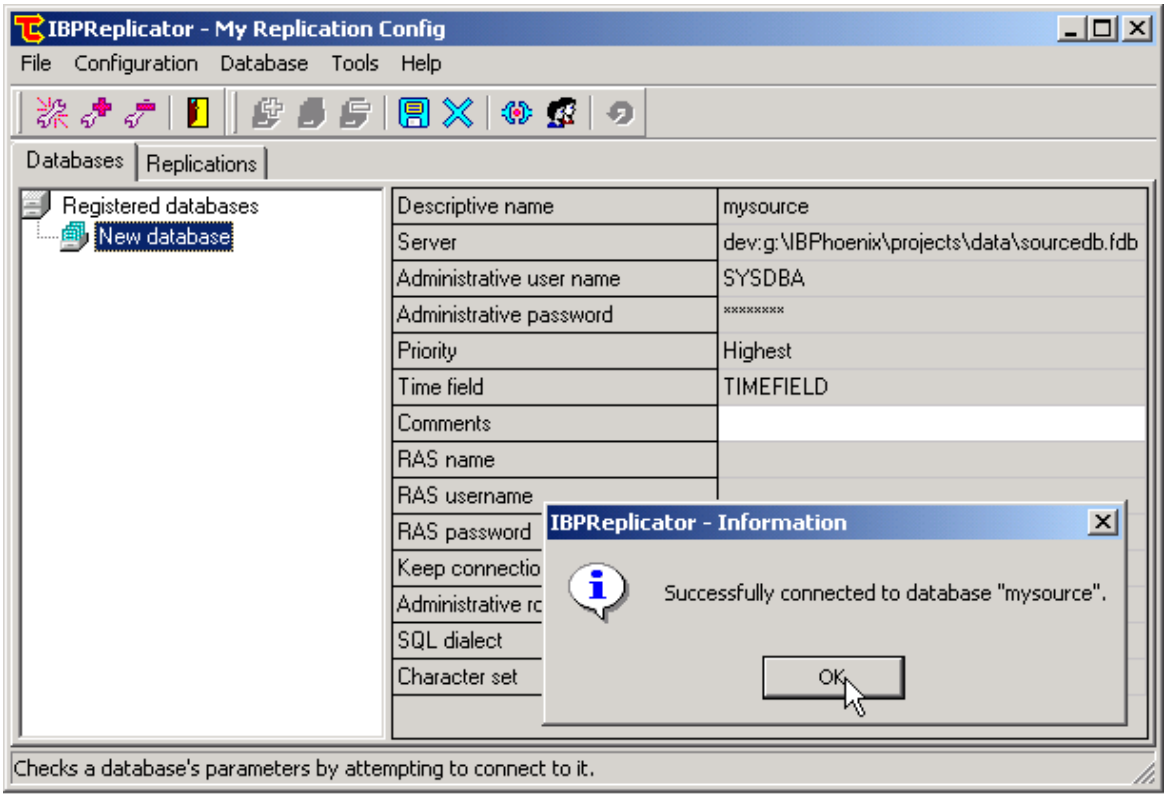
When you have defined the database, test the connection to make sure that you have input the details correctly. You can do this before saving the current parameters.

Right click on the name in the registration pane, or click the button on the toolbar:

© 2007 IBPh en Ed rs



If all is well, you should see this message:

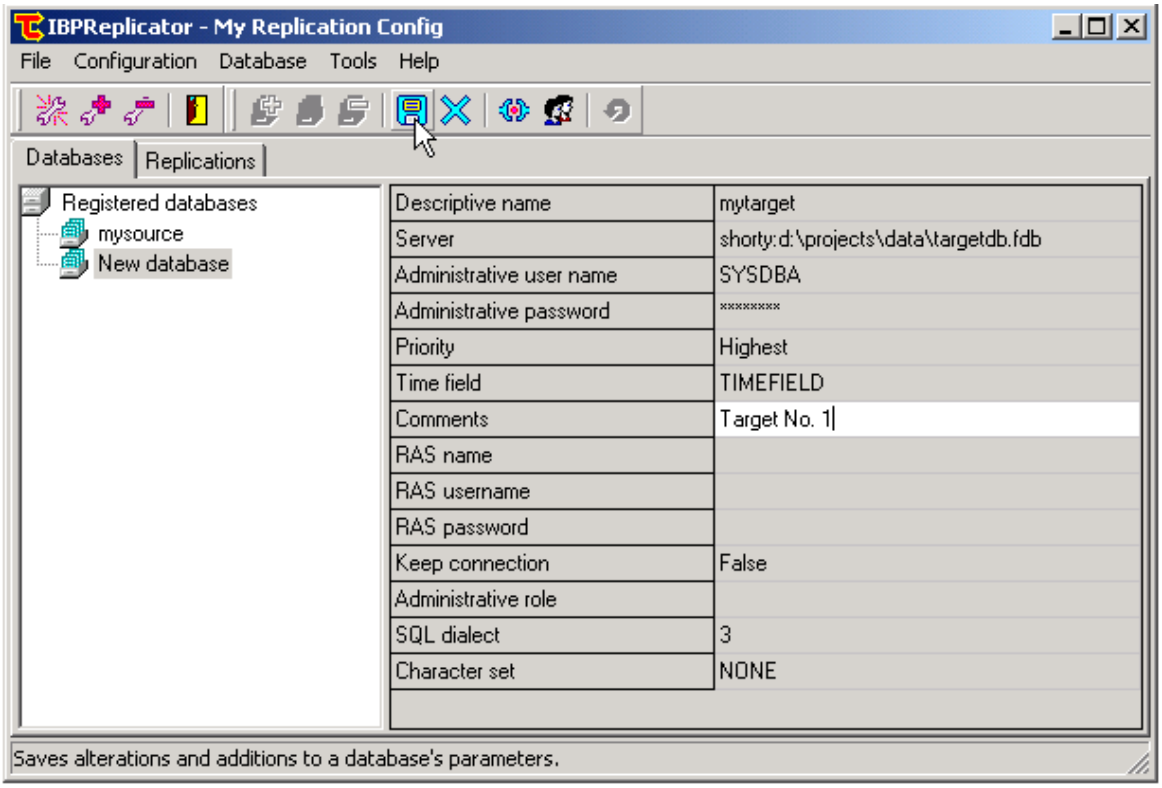


If the connection attempt fails, check and correct the parameters until the connection test succeeds.

Registering the Database

Finally, to register the database, just click the button. The newly registered database's friendly name is added to the list.

It takes at least two registered databases to make a replication configuration, so carry on and register each database as required.



Once you save the last of your databases, you are ready to move on and define a replication schema 42. Activate the Replications tab in the left-hand pane to open the page for this task.

6.2.1 About Licences

You can use the Replication Manager for many of your initial configuration tasks without needing to install your licences. However, you won't be able to start a replication server or activate any of your replications until you install your licences.

The Replication Server includes a single replicant licence, which will be counted when the

process checks that there are enough Replicant licences available for all of the databases defined to the current replication schema. Beyond replicating from one source server to one target, licenses are needed for all other databases that are involved. Licenses are entered using the License Manager dialog, which you access from the Tools Menu of Replication Manager.

If you have not purchased any licences, just start by using the EVAL license for testing a simple, one-way replication schema.

- For more details about IBReplicator licensing, see the topic in the IBReplicator Requirements 6 section.
- For instructions about installing licences, go to the License Manager 91 topic.

© 2007 IBPh en Ed rs

Chapter 7 - Defining a Replication Schema

Chapter 7 - Defining a Replication Schema

After registering all the databases that are to be involved in your replication schema, the next task is to identify the data that will be replicated.

Summary of Steps

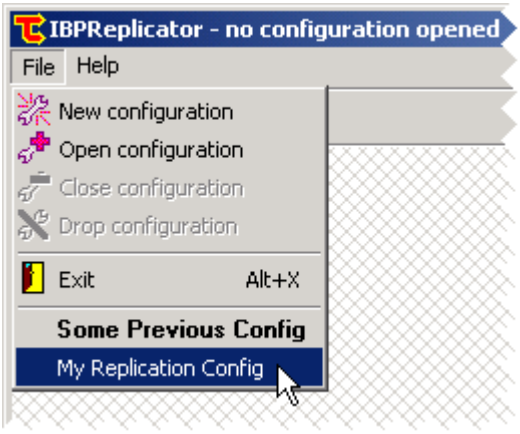
The steps involved are.-

1. Customising the default settings 43 (optional)
2. Creating the schema 45
3. Choosing the source database 50
4. Choosing the target database 54
5. Choosing replicated tables 58
6. Table settings 62
7. One-time synchronization 65
8. Identifying primary keys 69
9. Choosing replicated columns 71
10. Creating system objects in the source database 72

Assigning the Default Configuration Database

One configuration database must be assigned as the default. Once the Replication Server is running, it will automatically connect to your default configuration database and use the information there to do its work.

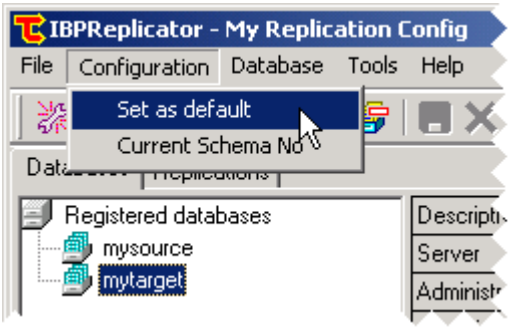
You can pick up the configuration you want to work with by clicking File on the Replication Manager menu and selecting one from the drop-down list:



If a default configuration database is currently assigned, it appears in bold-face. You can change the default one by selecting a different one from the list.

Select the configuration database you want to be the default and click to open it.

Now, select Configuration from the menu and simply click on "Set as Default":



Next

Your next step will be to create the Schema [45] for your replication.

However, you may wish to look over the default settings [43] first and decide whether you want to customise the options [43].

7.1 Customising the Default Settings

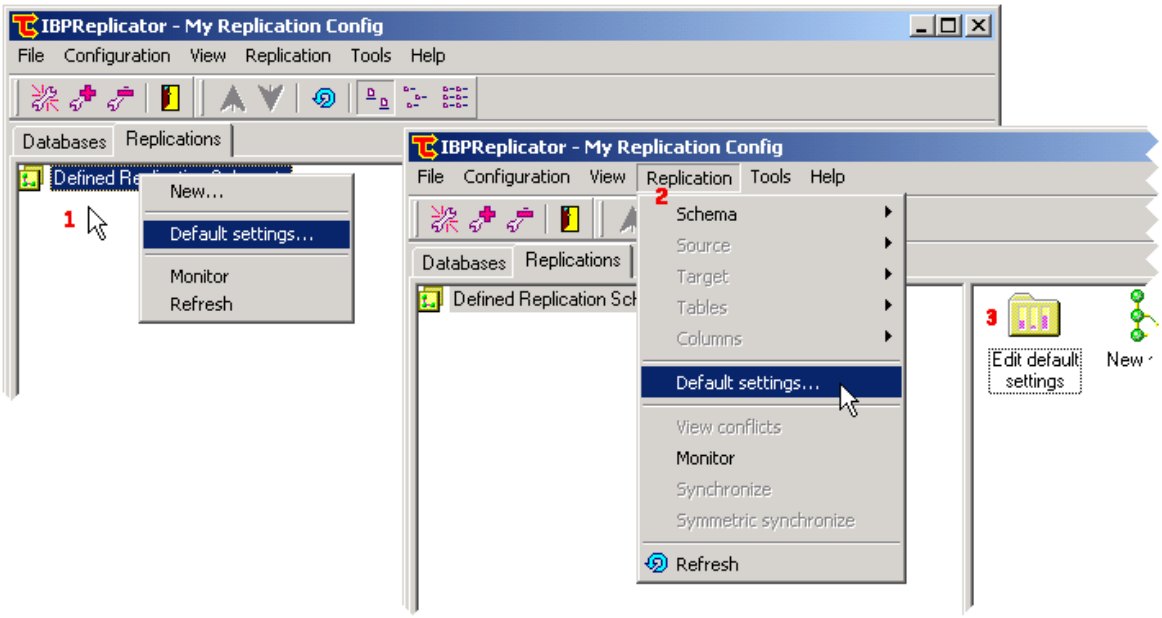
The default settings are stored in the configuration database when it is created. You can configure multiple schemata, each with its own default settings.

Changing the schema defaults does not change the settings of schemata that have already been defined; only newly defined schemata will have their settings initialized to the new values.

The settings for each schema you are going to define can be customised to

- control replication intervals
- set the conflict resolution strategy
- set the levels of logging you want for replication events

© 2007 IBPh en Ed rs



To begin modifying any of the default settings, select the root node labelled Defined Replication Schemata from the tree in the left pane. You can reach the input screen in one of the following ways:

1. Right click on the node and select Default settings... from the context menu
2. Access them from the menu via Replication | Default settings...
3. Double click on the icon in the right pane labelled Edit default settings

General Settings

If you are upgrading from a previous version of IBReplicator, you may notice that the default replication scheduling attributes no longer appear in this display. From v.2.5 onward, all replication scheduling is done in a new engine, within Replication Manager. Please refer to the revised notes about the Scheduler 86.

Conflict resolution strategy

When replicating new data from a source database to a target database where the primary key for the replicated row already exists, IBReplicator can be defaulted to use a specific strategy to resolve conflicts between source and target records:

Priority-based: When you register a database, you can give it a priority so that when conflicts occur, the database with the higher priority number takes precedence. If the source database has the higher precedence, the target's row is over-written, but if the target has the higher priority, its row is preserved and the replicated row is queued for manual conflict resolution.

© 2007 IBPh en Ed rs

Time-stamped: The more recent of the conflicting rows is preserved. This strategy requires the presence of matching timestamp fields in both source and target rows.

Master-slave: The source database's row always replaces the conflicting row in the target.

Conflicts that cannot be resolved directly by the conflict resolution strategy are logged for manual intervention using the Conflicts tool.

Event Logging

Each configuration database can have its own error/information logging settings and log file. Allowing IBReplicator to record its activity in either a window and/or on a log file on disk. The user defines what is written (level) and where logs are written.

The logging levels are

0: No Logging
1: Errors Only
2: Warnings and Detailed Error Description
3: Information
4: Detailed Information
5: Debug

Logs can be written to a disk file at either the default path (replicate.log in the IBReplicator installation directory) or at a different one, which you you specify.

> If you are setting up more than one replication you will probably want to specify non-default paths and/or file names for each one's log.

7.2 Creating the Schema

A replication schema contains information about what should be replicated, from where, to where. For the purposes of this help document, we'll assume that you are working through the steps previously described and are creating your first schema.

However, you can define multiple schemata for multiple replications that are to be cycled through in order. Typically you would already have registered databases, defaults, etc. already defined, and would wish to pick up definitions from some level of the "tree" to generate successive definitions in your new schema.

User Interface in v.2.5

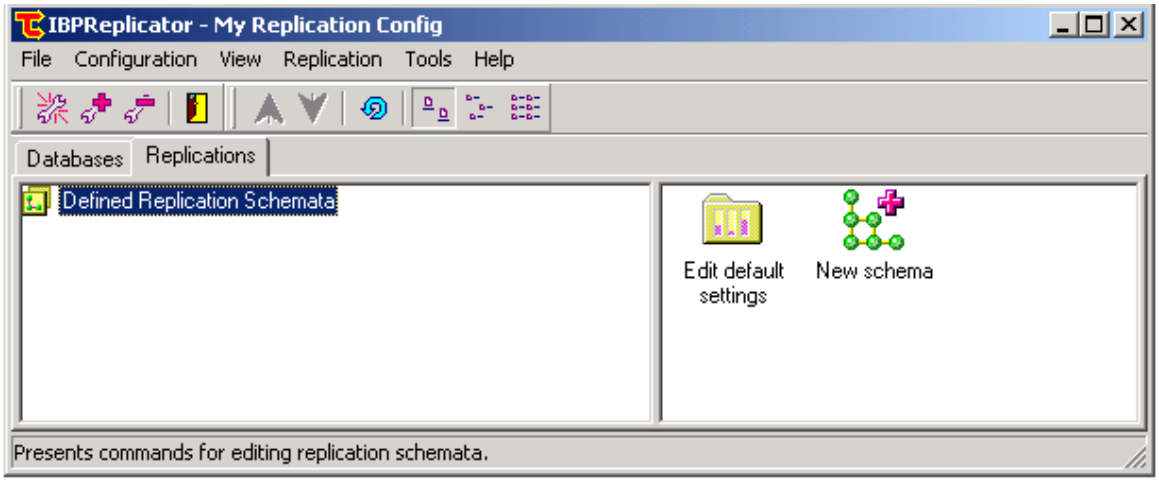
If you are upgrading to v.2.5 from an older version of IBReplicator, you will notice that the user interface has undergone some changes. Steps that may have been in pop-up dialogs previously have been integrated into a single schema definition panel.

Another change that comes with v.2.5 is that you can customise the order in which multiple schemata will be cycled through. The instructions are at the end of this topic.

© 2007 IBPh en Ed rs

The Replication Tree

The replication tree--the left pane of the Replications tab in Replication Manager--provides the details of everything you have so far defined in the currently selected configuration database:



At this point, the illustration above does not show much at all! As you progress through your definitions, the tree will expand and enable you to access aspects of your schema at each level of detail.

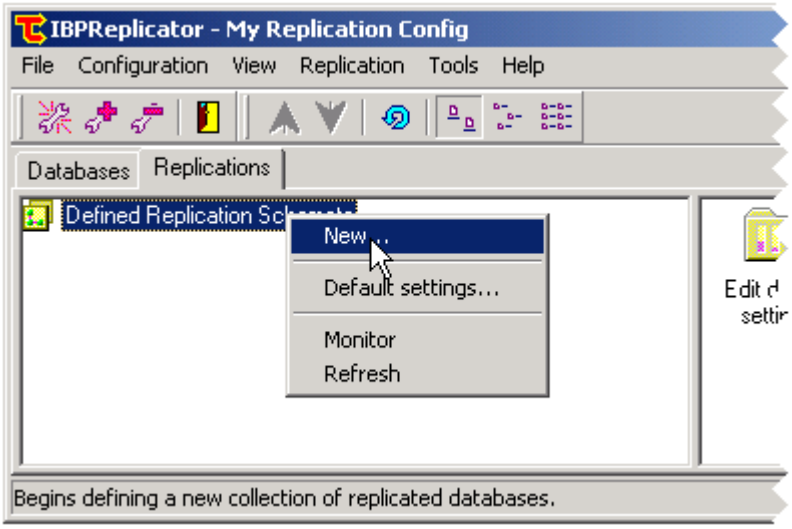
Ways to Work on Definitions

The interface provides several different ways you can approach definition and maintenance tasks at various levels:

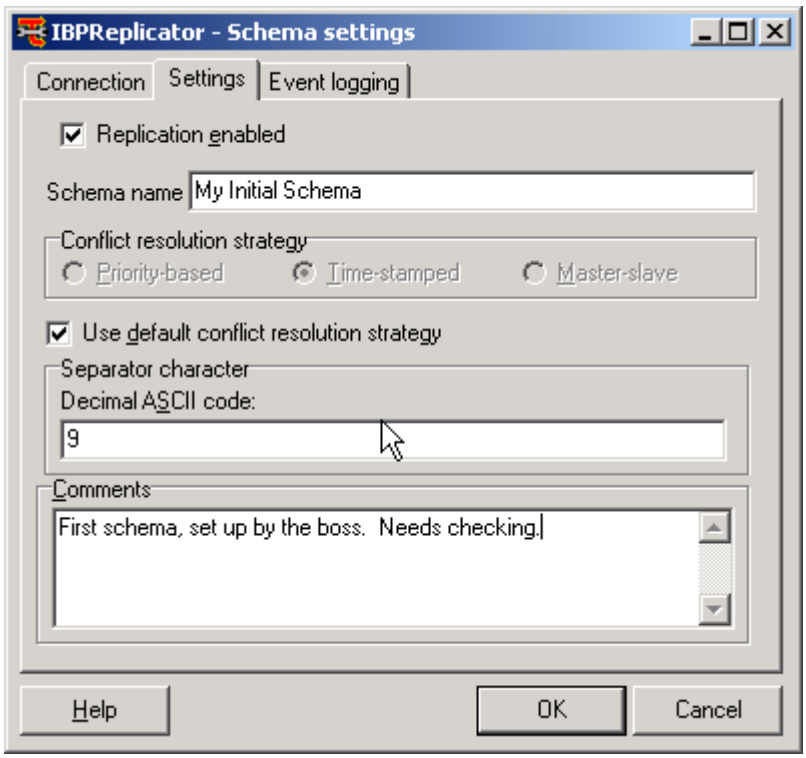
- using the right-click context menus on the tree nodes,
- right-clicking on the icons in the Tasks pane to display the context menu for a task
- double-clicking on the icons in the Tasks pane
- making selections from the Replication menu.

You can mix the tricks in whatever way suits you best. Following are some examples of the various approaches.

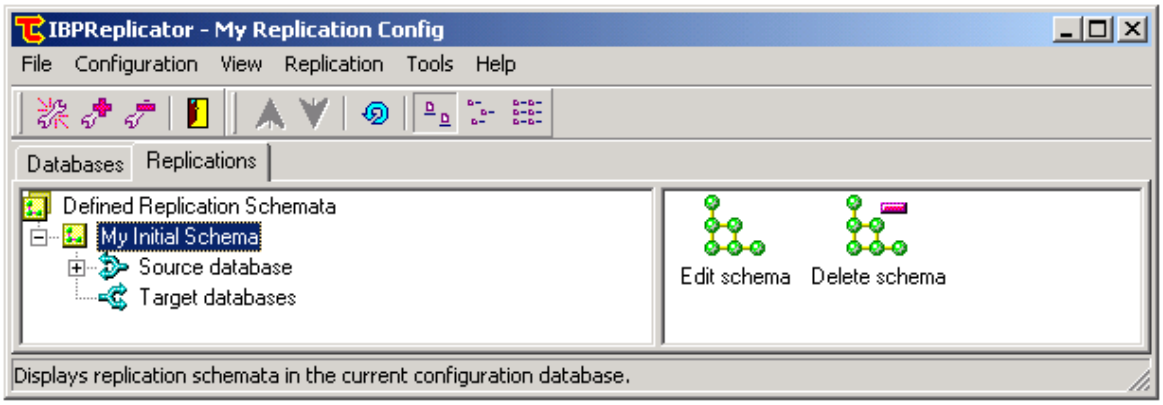
© 2007 IBPh en Ed rs



Here, the root node of the replication tree is selected and the user right-clicks on it to display the context menu. The New... option is selected and a left click will provide the input dialog for defining a new schema:



After you click OK, the new schema appears in the tree with nodes for the source and target databases:



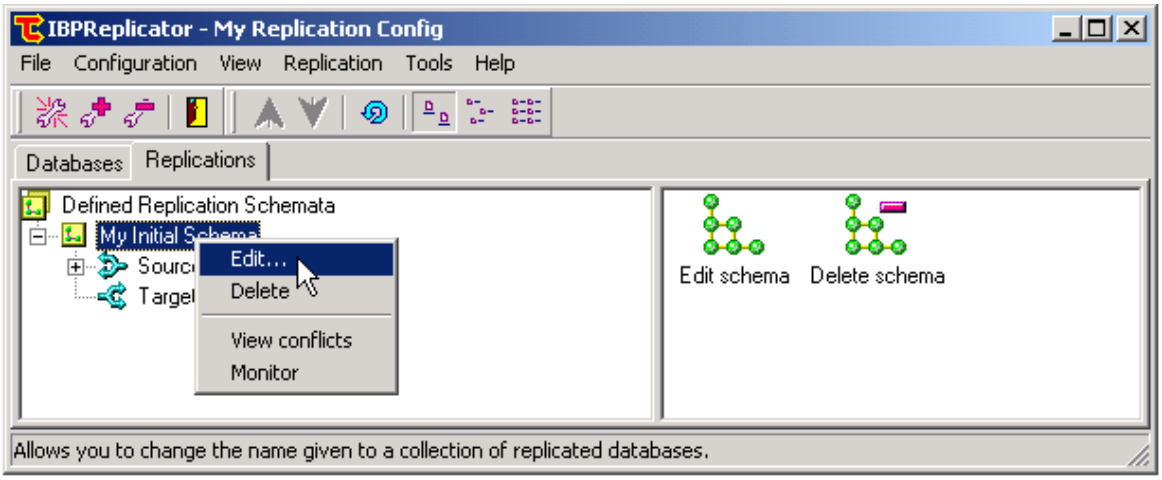
As another approach, you could have double-clicked on the "New schema" icon in the Tasks pane, or you could have right-clicked on the same icon to reach the New.. dialog from the icon's context menu.

> TIP Use schema names that will be meaningful not just to you but to anyone else who is going to manage your replications!

Note that you could have stayed within the schema definition dialog to develop the schema definition further, as described below, clicking OK once you were ready to save the work.

Tasks Pane Context

Notice how the icons in the Tasks pane change, according to where the focus in the tree pane is. For example, if you single-click the newly created Schema node ("Another Schema"--an entirely useless name for a schema, incidentally!) you will see new icons in the Tasks pane enabling you to edit or delete the selected schema:



As with other nodes, you also have similar contextual options by selecting and right-clicking the schema's node directly in the the tree. In each approach, you will be able to edit the name of the schema or delete it entirely.

> Be aware that deleting a schema will remove all replication information for that schema, including anything stored in lower nodes. Once the schema is gone, the

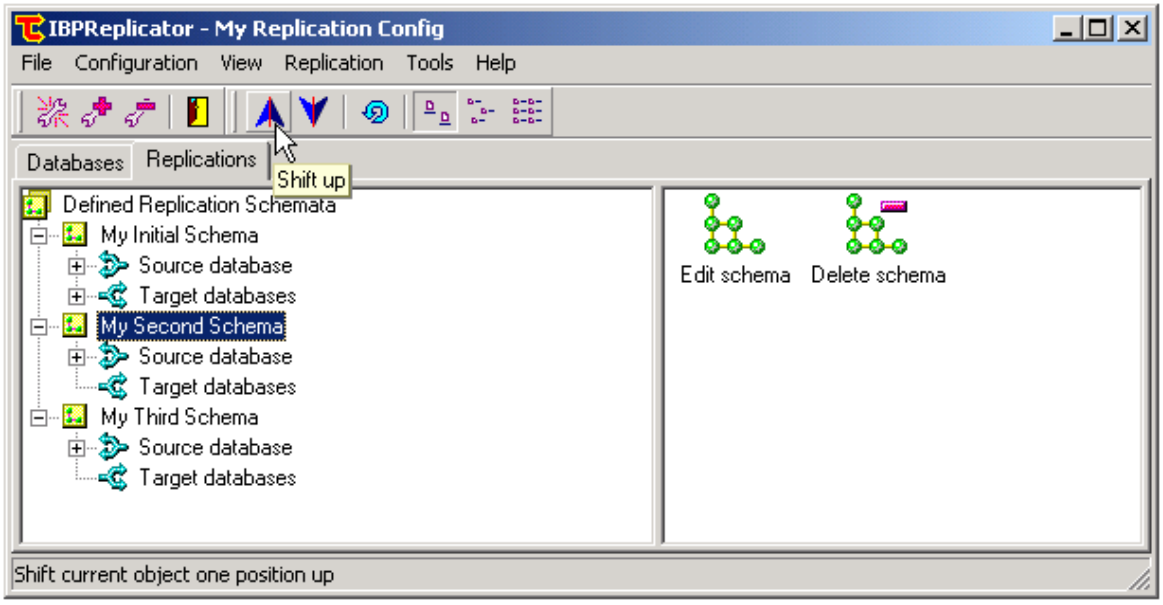
replication cannot occur!

Expanding the schema node reveals two new nodes labelled source database 50 and target database 54. These nodes can be approached in the succeeding steps.

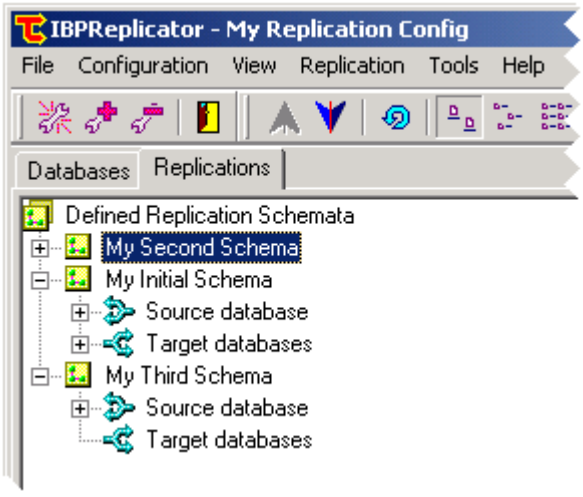
Adding More Schemata and Ordering Them

To add another schema, simply select the Defined Replication Schemata node again and repeat what you did for the initial one.

Once you have more than one schema, the Shift Up and Shift Down buttons are enabled in the toolbar when a schema node is selected. They provide the means to alter the order in which your replications will take place in the cycle:



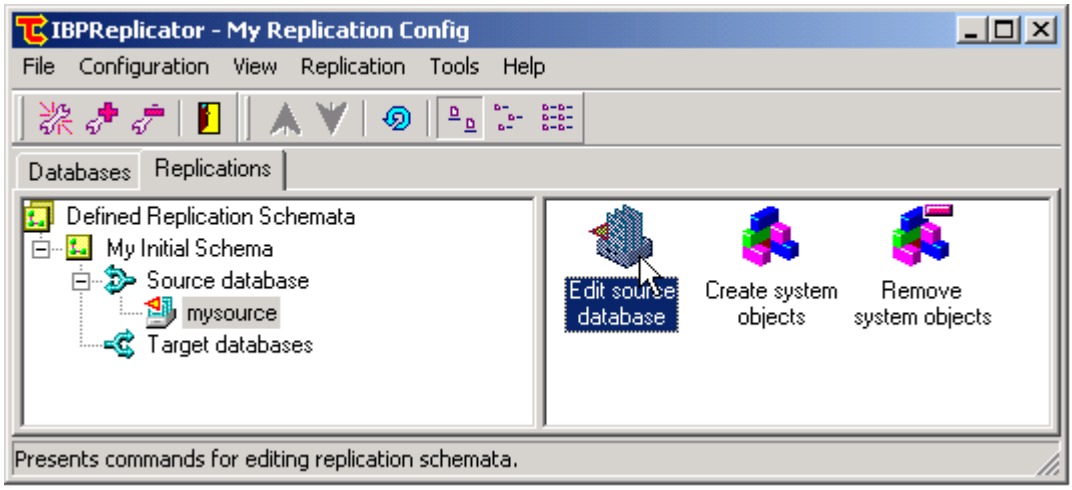
Click on the Shift Up button to move the selected schema up the order or Shift Down to move it down:



7.3 Choosing the Source Database

The source database is the one in your replicant pair that will be replicated from. You already defined your source database at an earlier step, when registering databases 33.

Expand the schema node and select the Source database node. Use one of the methods described in the previous section 46 to open the Edit source database dialog:



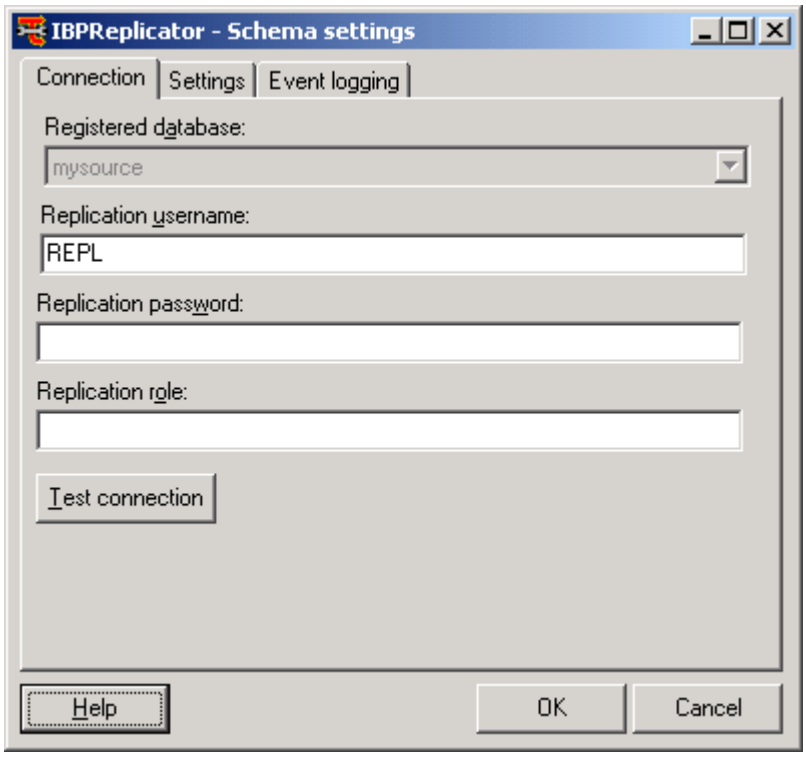
Connection

The first tab of the dialog shows the database that will be the source of your replication.

Registered database

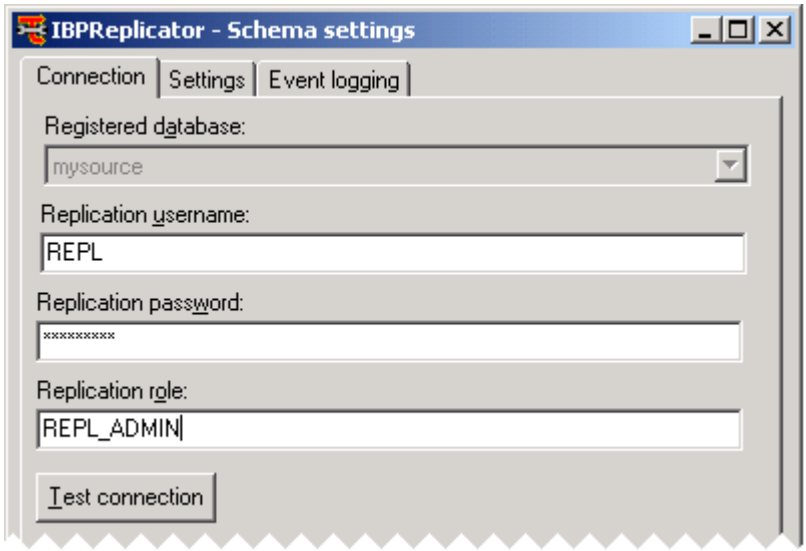
In the example, the free text descriptions of some databases registered in the previous examples appear in the Registered database field. Because there can be only one source database for a replication, the "friendly name" of your source database is greyed out and not editable:

© 2007 IBPh en Ed rs



Username, password and role

Supply a username, password and role to be used by the replication server when it connects to the source database:



The name of the "replication user" does not have to be the SYSDBA or that of an Administrator of the operating system. The default user provided in the dialog is REPL, which can be used for any replication and must be used if you plan to implement bi-directional or n-way replication.

About the REPL user

The user named REPL is recognised by IBReplicator as a special user. Any changes made to the source database by the user REPL will NOT be replicated. This is by design, to

prevent replications from a target database back to a source from going into a loop.

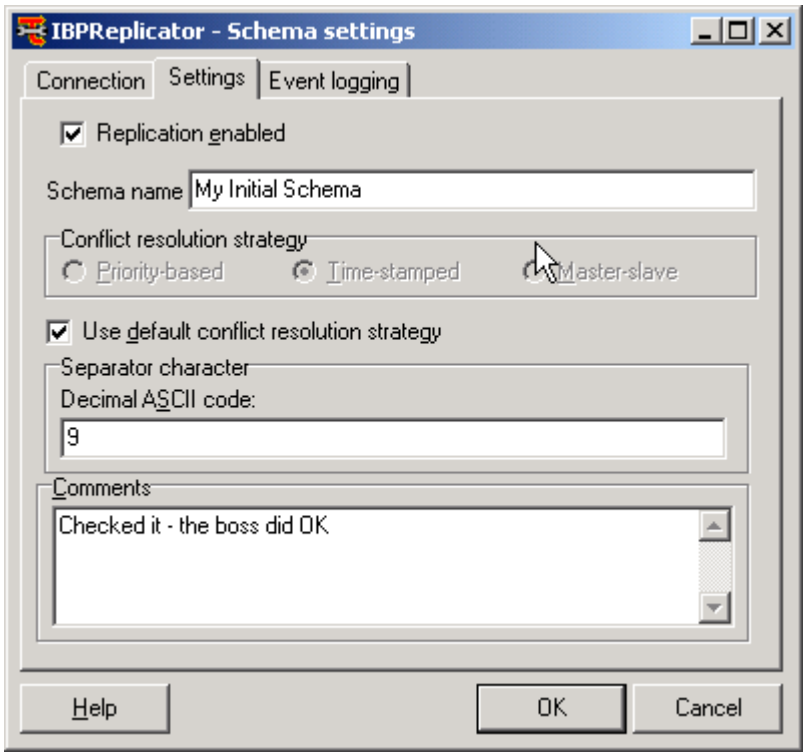
Test the connection

Whatever user you assign, make sure that it is added to the relevant Firebird or InterBase security databases on each machine involved in the replication. Once you have done that, click the Test Connection button, enter the username and password (and the role, if you have defined one) into the login dialog. If login is unsuccessful, review your entries and make any necessary corrections before trying again.

> It is pointless proceeding now unless you can make a successful connection test.

Settings

The second tab enables you to customise settings for this source-target pair. They are initialised to the default settings, but can be modified for each source database, without affecting either the defaults you have already set or any settings you might have customised for another replication.



You can add any free-form comments you like in the Comments box, to augment or override any comments inherited from the defaults.

Replication enabled

This check box should be checked ON unless you need to prevent replication from occurring. For example, you would check it off if you were taking the network down or powering one of the servers down for maintenance.

Conflict resolution strategy

The replication is initalised to use the default conflict resolution strategy that you might have customised previously—refer back to Customising the Default Settings 43. The current default setting appears above, greyed out. In the illustration it is Master-slave.

You can override the default conflict resolution strategy by checking off Use default conflict resolution strategy and specifying one.

> Don't select Time-stamped unless you have the necessary Timefield columns in all of your source and target tables.

Separator character

IBReplicator stores the primary keys in text format. The separator character is used to separate the values in primary keys with multiple columns. Thus it is essential that this be a character that cannot appear in a key value.

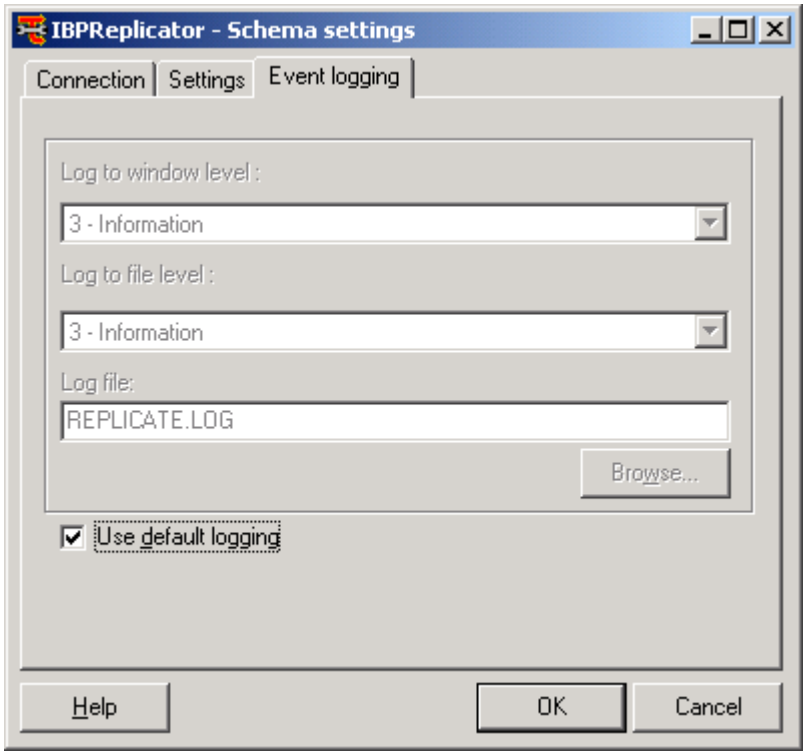
The default character for v.2.5 is ASCII 9, a non-printable character sometimes represented by the musical "minim" symbol or Ctrl-I. In the unlikely event that ASCII 9 would conflict with values in your data's primary keys, provide the code for one that is suitably improbable.

Comments

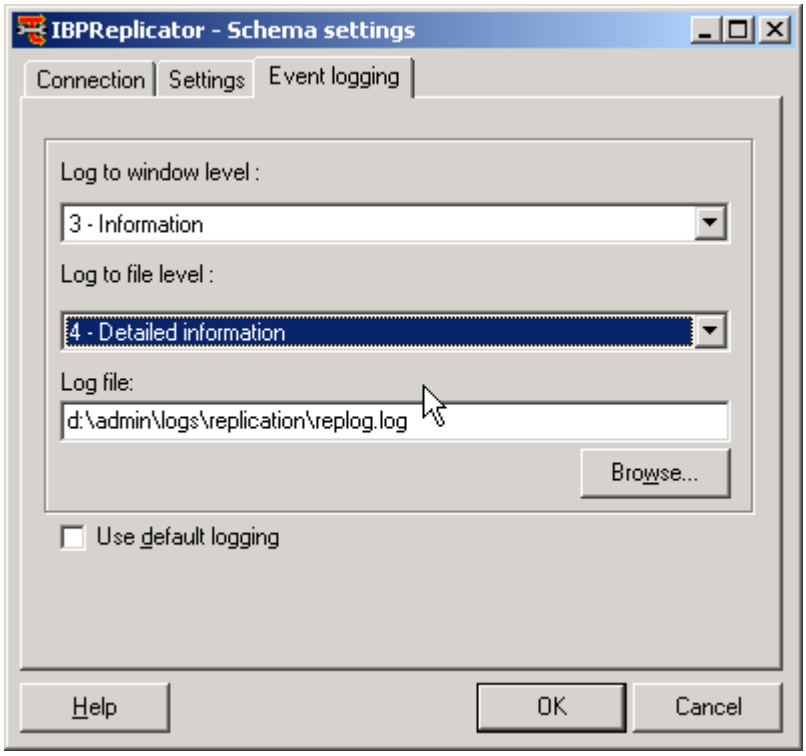
The comment field is entirely for your own convenience. Use it for any form of documentation that is useful to you.

Event Logging

Use the default logging strategy, or specify a specific logging strategy for this source target pair. A newly configured source database is initialised to use the server defaults:



If you are defining multiple replication schemas, you might wish to have the server log each one to its own log file. In that case, check off Use Default Logging and enter a path that exists on the filesystem of the IBReplicator host machine and, if needed, a different log file name that can be identified easily by the system administrator:



Click OK to update the details of the source database node.

Now, you will be able to select a source database node for editing the source database details, removing the source database and accessing the system objects for that source database.

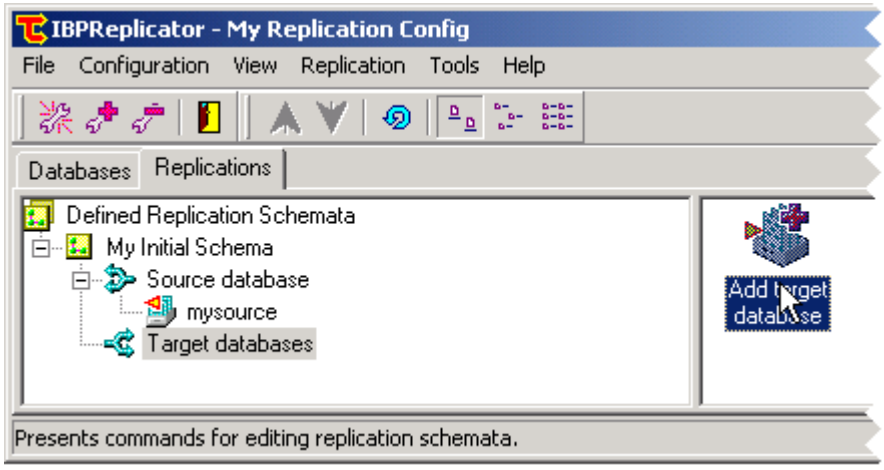
The commands for creating a removing system objects are discussed in Creating Source System Objects 72.

Each schema can only have one source database, so you need to define a schema for each of your replications. A source database in one schema can be a target database in another schema, and a single source can replicate to multiple targets.

7.4 Choosing the Target Database

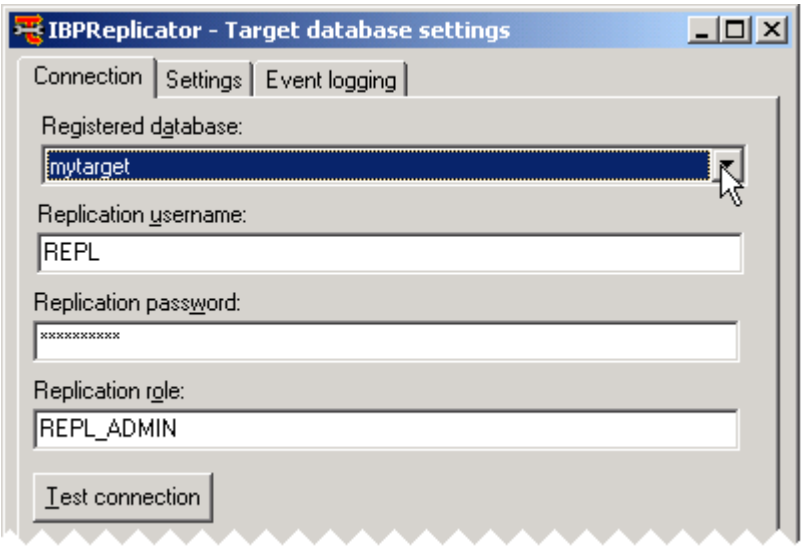
A target database is a database to which replicated data is sent. When identical data is to be sent to multiple target databases it possible to define an initial target database and then copy (clone) that definition and apply it to other target databases.

To define a target database, expand the schema node of your schema in the tree and select its Target databases node. Use one of the methods previously described 46 to select the Add Target task.



Target Database Settings

Connection Tab

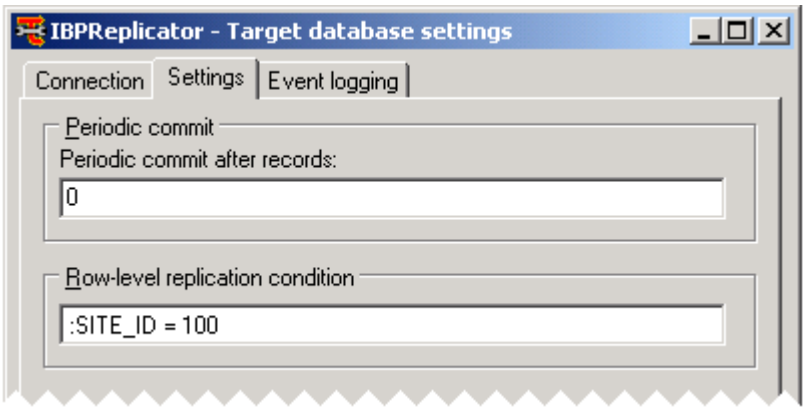


- Drop down the list of registered databases and choose the one that is to be the target database.

- Supply the username, password and role to be used by the replication server when it connects to the target database. The name of the "replication user" does not have to be the SYSDBA or that of an Administrator of the operating system. The default user provided in the dialog is REPL, which can be used for any replication and must be used if you plan to implement bi-directional or n-way replication.

Settings Tab

© 2007 IBPh en Ed rs



Periodic commit

Ordinarily IBReplicator commits changes once per replication, after all the logged changes have been successfully replicated to the target database. If a network error occurs during replication, all of the changes will be rolled back and IBReplicator will try again.

If the network connection is unreliable, or the number of changes to be replicated is large, you can specify for changes to be committed periodically as the replication proceeds.

In the field labelled Periodic commit after records set a positive value representing the number of successfully replicated rows that IBReplicator is to group and commit together.

> CAVEAT Don't implement this option unless it is actually necessary. It can impact the general performance of the database over time and breaking a replication into numerous separate transactions breaks the overall atomicity of a replication.

Row-level replication condition

This is an optional setting that allows rows to be selected for replication on the basis of some value in the data.

For example, if you have defined a multi-site database with conditional replication in view, you might have a key on some tables that identifies the site when the data in a record pertains to that site and not to other sites in the schema. This setting can define this condition.

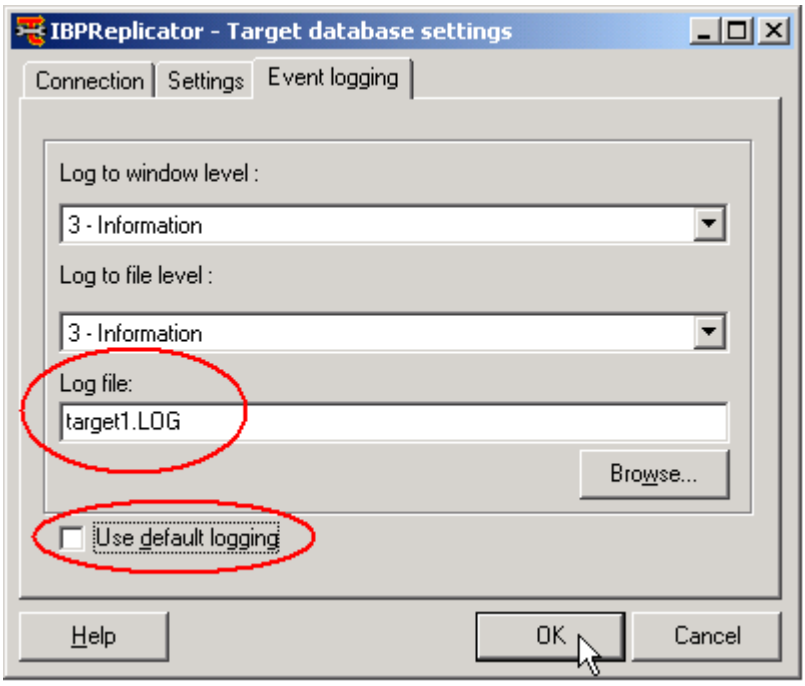
To implement this, any tables that you want to apply this global condition to should include a column with an identifier that is the same for all such tables. Express the condition like a predicate inside a WHERE clause.

In the example illustrated the condition :SITE_ID = 100 will tell IBReplicator to look out for columns named "SITE_ID" in the replications it performs to this target and ignore any rows that have any value other than 100 in this column.

Row-level conditioning can also be done at individual table level. For details and more information about the row-level conditioning expressions to use, look here 64.

Event Logging Tab

In v.2.5 and above you can optionally override the default log file and define a specific log file and logging levels for this particular target database:



Check off "Use default logging" and adjust the event logging settings to suit your needs.

When you are done, click OK to add the details of the target database node to the tree as a child of the Target databases node.

Accessing the Node Again

In the following sections, you can find out how to

- select and map the tables and procedures 58 for the replication
- identify the primary keys 69
- choose which columns are to be replicated 71
- tell the Replication Manager to create the system objects 72 in the source database

Whenever you need to access a particular target database again for these tasks, or you want to edit or remove the target, just select it in the Target databases node and use one of the previously described methods 46 to select the task you want.

Cloning Targets

A schema can support one or multiple target databases for a source database, and different data can be replicated to different targets—see Choosing replicated tables 58 (next) and Choosing replicated columns 71. A separate definition of the tables, primary keys, columns and rows to be replicated is required for each target database involved. That makes it possible to have very fine-grained control for any schema.

It is common for a schema to encompass a number of targets that are going to be replicated with the same, or very similar, data. Because setting up numerous such target databases manually could be repetitious and time-consuming, it is possible to set up one target it and clone its definitions to other targets.

Cloning a target can be done several ways, much as you do with other tasks already discussed. You can

- drag the specific target database node that you want to clone and drop it onto the general target database node or another schema node
- select the specific target database, right click and choose Clone from the context menu
- use Replication | Target | Clone from the replication menu

Once the new target is in the tree, supply the name by selecting from the list of databases that has been registered. Click OK and a copy of the schema defined for the original target database will be created for the cloned target database. Any differences required can be tidied up by editing the cloned definitions.

> TIP Perform any cloning once you have defined most or all of the mappings for your "base" target.

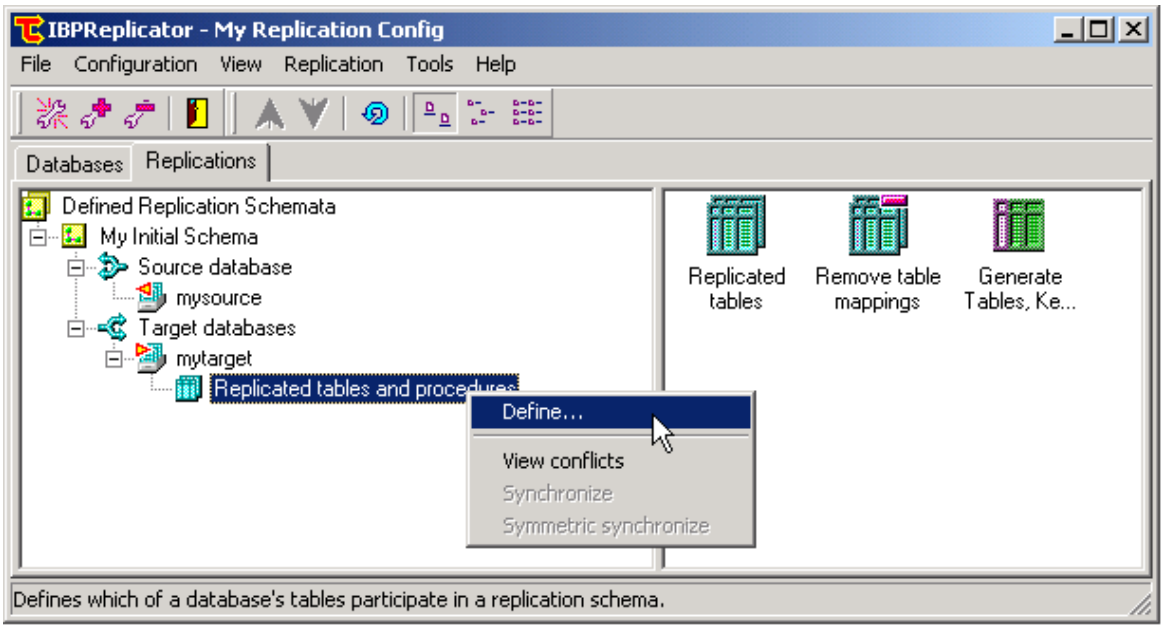
7.5 Choosing Replicated Tables

Replication is essentially a matter of copying data from tables in the source database to tables in one or more target databases. This step is the key one in defining your replication schema: deciding which source tables you want to replicate in your target[s].

It is probably stating the obvious to remind you that you do need to start out with both a source database and a target database that already have at least the tables you want to replicate defined in their database schemata!

> Don't forget the two essential metadata attributes needed in tables you are going to be replicating from and to: they must have a primary key and, if you intend to use timestamping for conflict resolution, both the source and target tables must have matching TIMESTAMP columns (or DATE in IB 5.x databases) that can be referred to as an age reference.

© 2007 IBPh en Ed rs

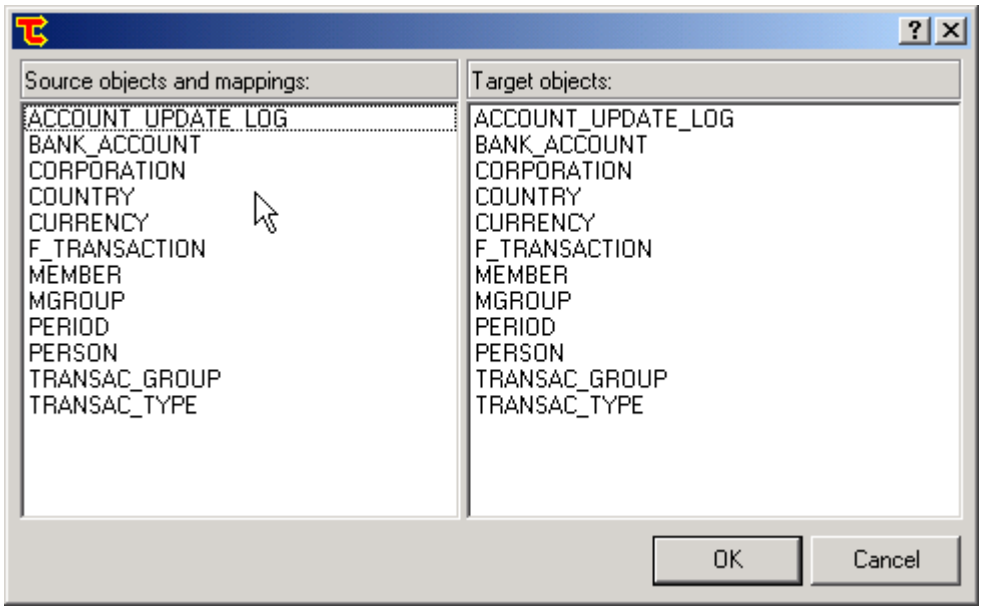


Expand the node for a specific target database and select the Replicated tables and procedures node.

To open the Replicated tables and procedures dialog, either

- right-click and select Define from the context menu; or
- double click on the Replicated tables icon; or
- select Replication | Tables | Define from the Replication menu.

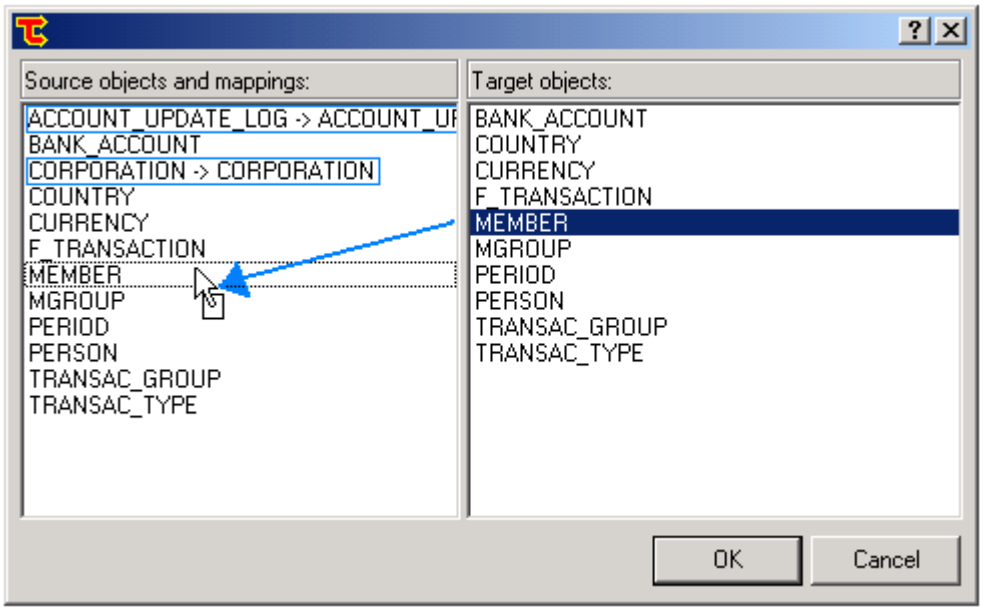
The Mapping Panels



© 2007 IBPh en Ed rs

Mapping the Tables

Now map the tables in the source database by selecting and dragging tables from the Target objects panel to the Source objects and mappings panel:



- You can multi-select more than one table or procedure (using shift-click or ctrl-click, as required) and drag them as whole
- Alternatively, just double-click on the target object to set the mapping(s) automatically between objects with identical names
- Dragging from the source database to the target allows you to break an already established mapping

Mapping in Bulk

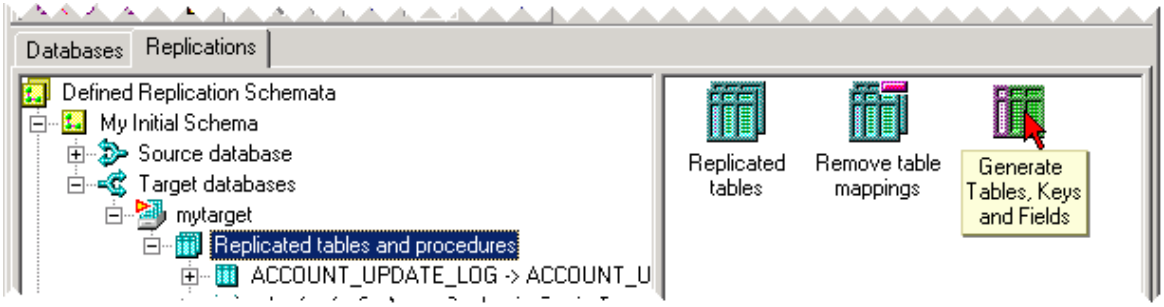
The mapping of tables, primary keys and fields can be done automatically by

- double clicking on the Generate Tables, Keys and Fields icon; or
- selecting Tables | Autogenerate from the Replication menu

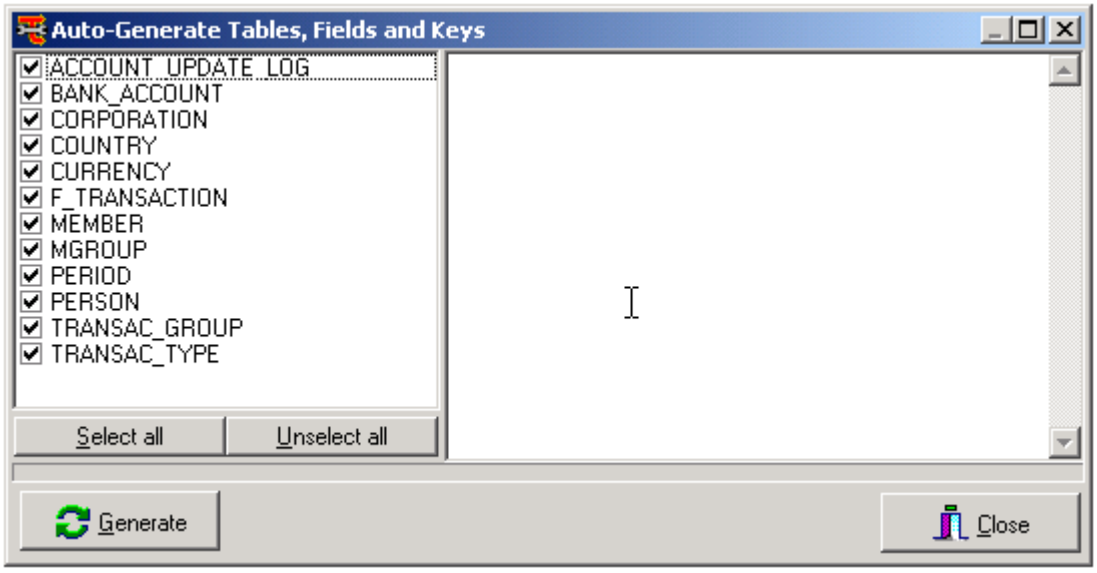
This choice causes all tables, keys and fields in the source database to be automatically mapped to corresponding objects of the same name in the target database. Previous mappings will be respected.

> From v.2.5 on, you can multi-select a selective group of tables and autogenerate the mappings and objects for them.

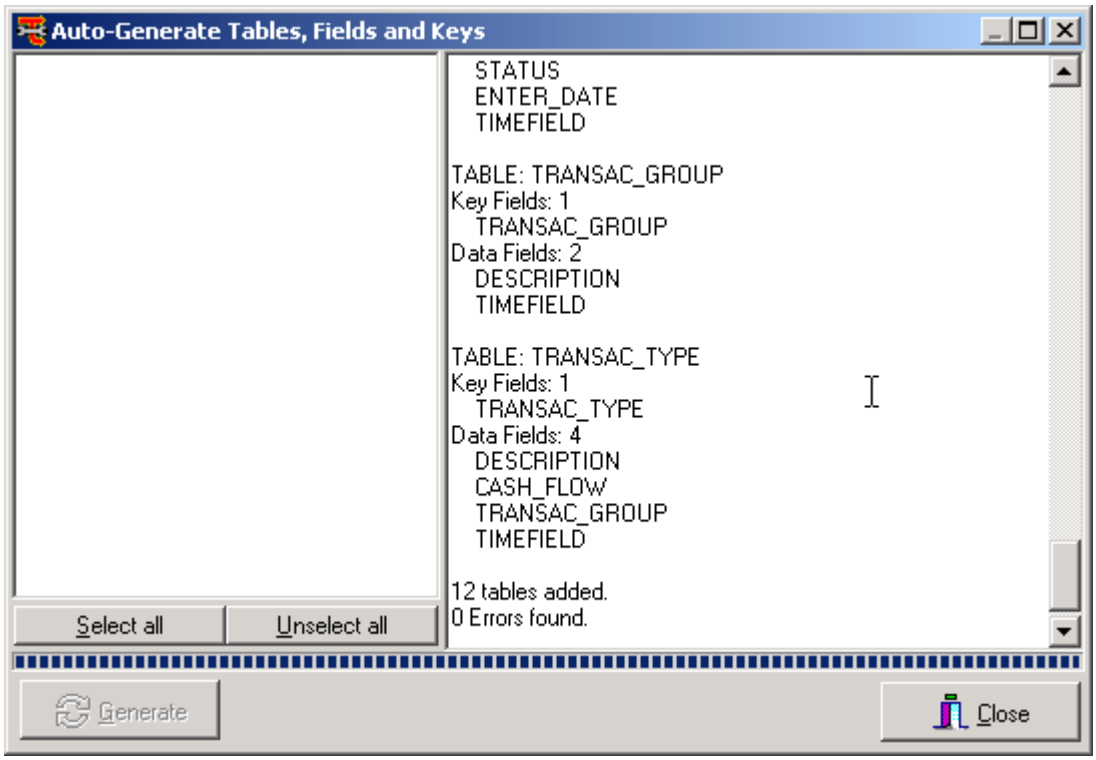
© 2007 IBPh en Ed rs



The Autogenerate dialog appears with all of your unmapped tables selected:



Check off any tables you do not want to autogenerate objects for, e.g. because you do not want to replicate them, and click the Generate button...



...and it is done!

Removing Mappings

To remove replicated tables, primary key and column definitions, select the table node, then either

- right-click and select Remove from the context menu; or
- double-click on the Remove table mapping icon; or
- select Replication | Tables | Remove from the Replication menu

Heterogeneous Mappings

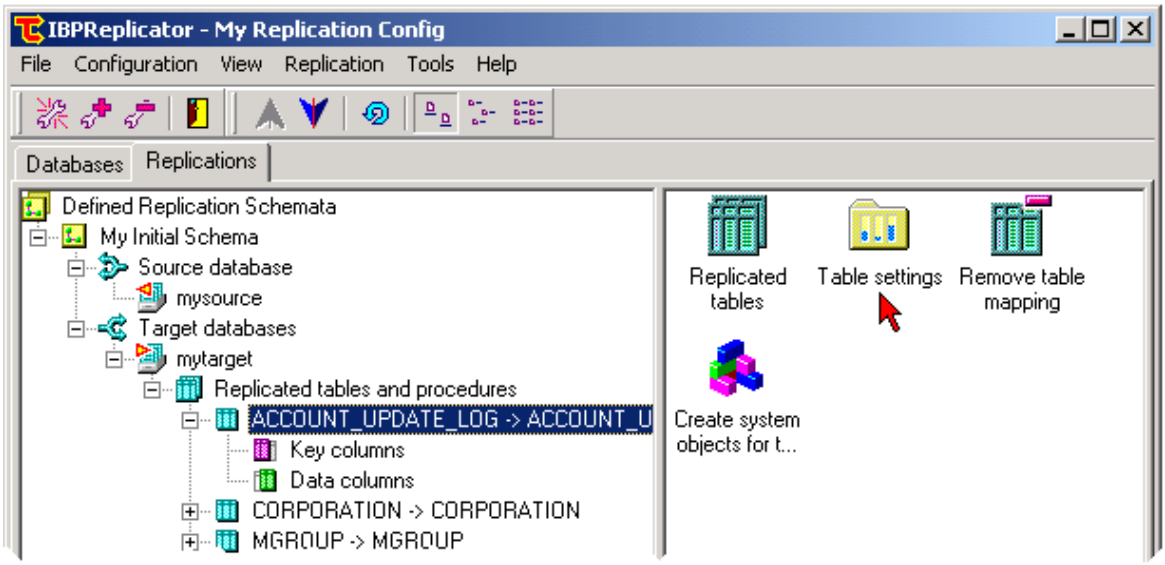
It is entirely feasible to create heterogenous table mappings where source and target tables have different table names and even if they have the same table names but different column names. However both tables must have unique primary keys that identify the rows to be replicated and that the columns being replicated have compatible data types.

7.6 Table Settings

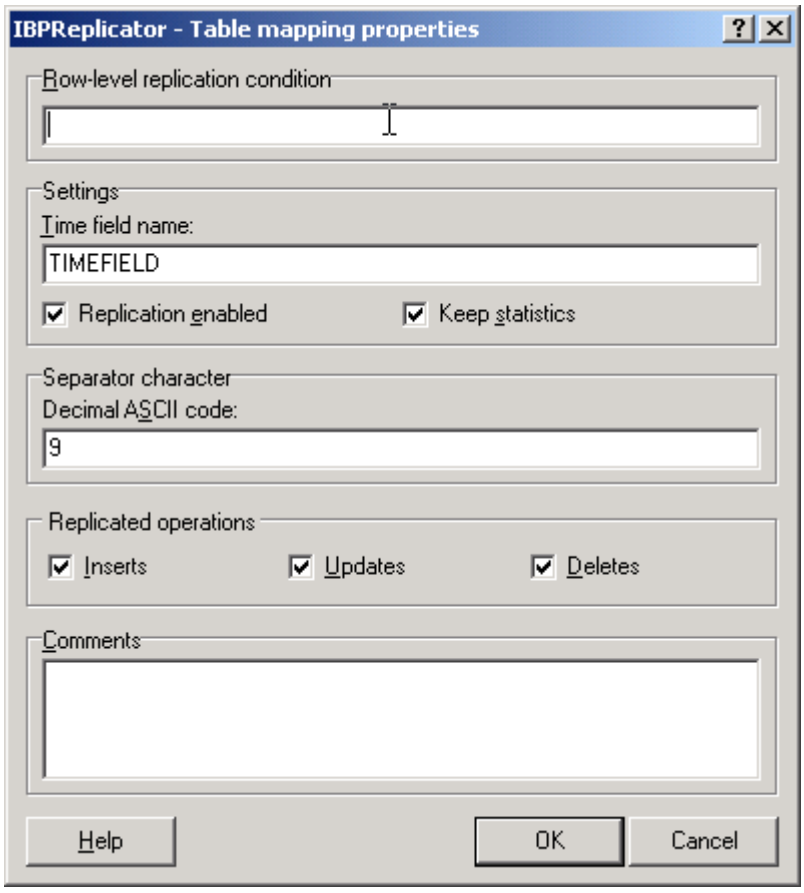
The Table settings dialog allows you to change the default settings for your source-target mappings for an individual table.

To access this dialog, select the table's node in the tree and either

- right click and select Settings; or
- double-click on the Table settings icon; or
- choose Replication | Tables | Settings from the Replication menu.



The Table Settings Dialog



Fields on the Table mapping properties Dialog

Row-level replication condition

This is a row selector, if you should need to limit the rows to be replicated by providing a WHERE clause.

Given the basic syntax of a SQL select expression:

```
SELECT <column list> FROM TABLE
  WHERE <some_condition>
```

you might, for example, request all unpaid invoices using

```
SELECT * FROM SALES WHERE PAID = 'N'
```

Say you want to replicate only rows that represent unpaid invoices. You would pick up the <some_condition> part of the WHERE clause and enter that into the row level replication condition as

```
:PAID = 'N'
```

Note the colon prefixed to the column name.

Although this example is simple, there is no limit to the complexity of the condition: any condition that would be valid in a normal IF(..) statement in PSQL can be used as a replication condition.

Time field name

If you are using the time-stamped conflict resolution strategy, both the source and target tables need a timestamp column from which IBReplicator is to read for comparing the ages of the rows to resolve conflicts. It should be a TIMESTAMP type (DATE type in IB 5.x) into which a regular Before Insert or Update trigger (separate triggers for InterBase) writes the system timestamp [CAST ('NOW' AS TIMESTAMP)] whenever an insert or update occurs.

IMPORTANT The time field must have the same name in both the source and target tables.

The Time field name is where you name and map this column for the two tables.

Replication enabled

For future use, unchecking the box will exclude this table from replication without removing the mappings. (Currently not implemented.)

Keep statistics

Check this box to use the Monitor tool to review IBReplicator's performance.

© 2007 IBPh en Ed rs

Separator character

Use this if the separator character for the elements of composite primary keys in an individual table need to be different from the global separator character 53.

Replicated operations

By default, all operations (insert, update and delete) will be replicated. You can choose to exclude an operation by checking it off.

Comments

Use this box for any free-form documentation that is useful to you.

7.7 One-time Synchronization

Synchronization allows the IBReplicator to get the source and target tables into the state where data matches in both, i.e. a one-time "global" replication.

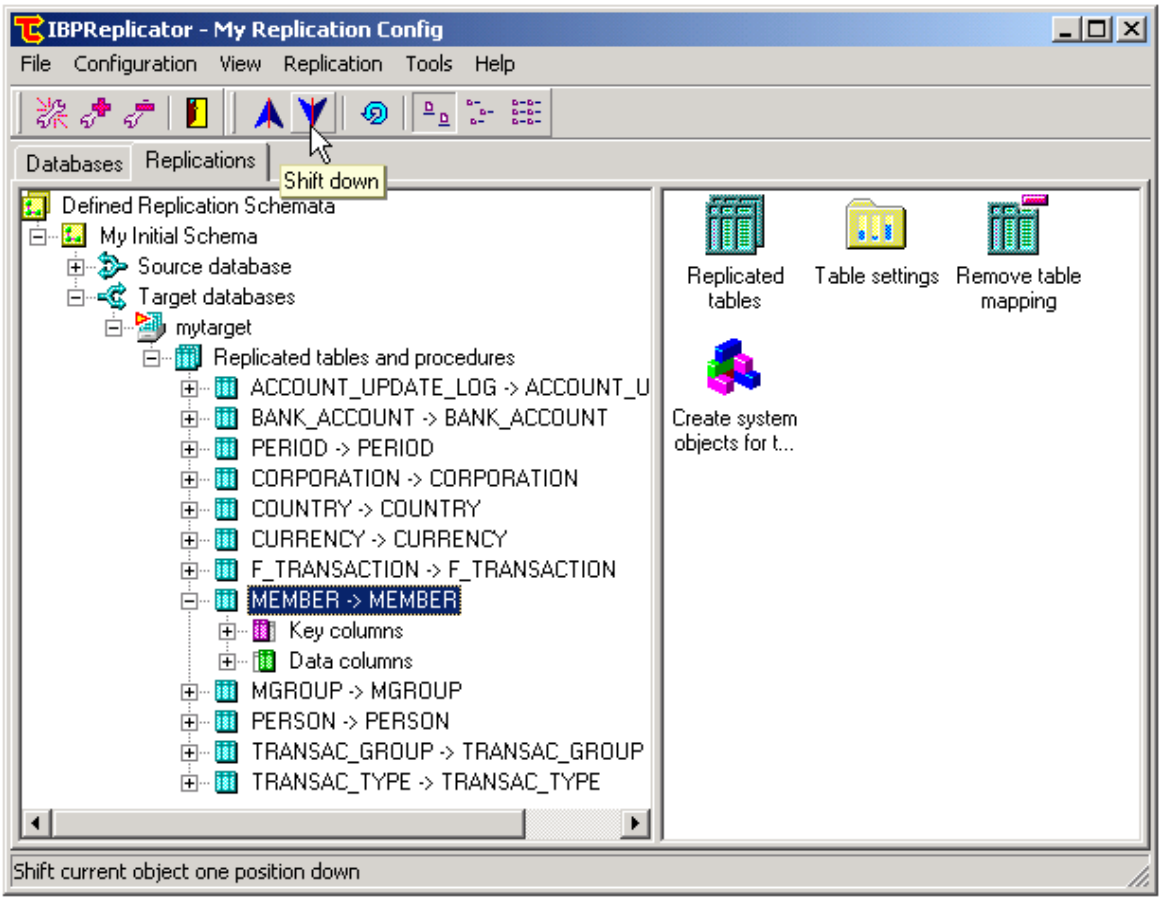
There are two types:

- asymmetric, where the target table is synchronized from the source table. The target table becomes an exact copy of the source table with IBReplicator performing all the necessary inserts and deletes.
- symmetric, which does not perform any deletes but, instead, inserts any records missing on the target from the source and vice-versa. It also performs updates according to the current conflict resolution strategy.

Customising the Order of Tables

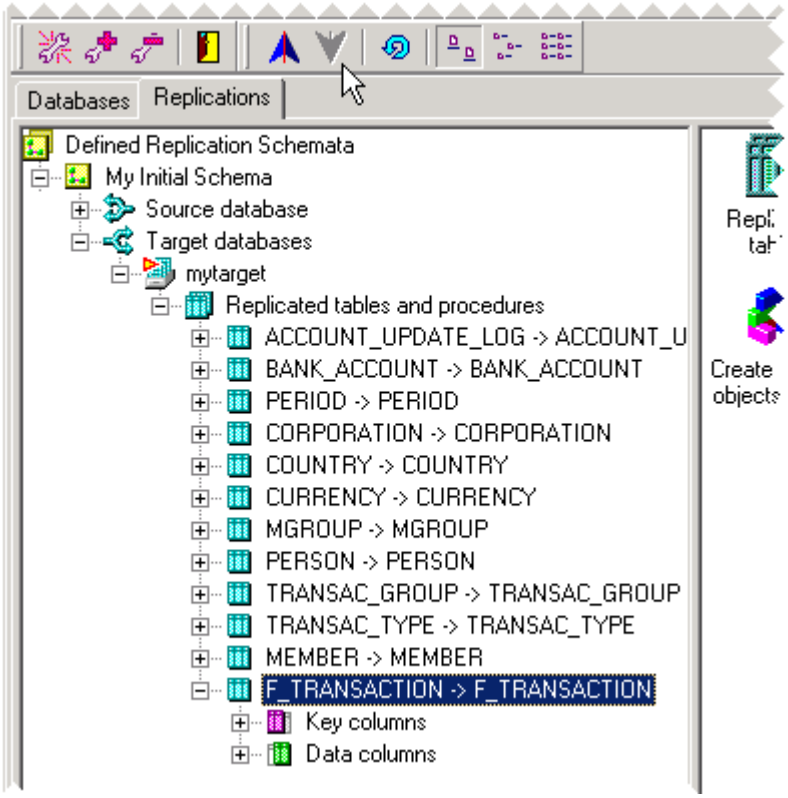
Prior to v.2.5, it was necessary to synchronize tables one-by-one, in order to avoid dependency conflicts when a bulk operation followed the default (alphabetical) order in accord with the system tables. Now, you can define your own order for the synchronization and perform it for all tables.

Select the Replicated Tables and Procedures node of the target database and select a table that you want to move. Then click either the Up or the Down button in the toolbar, to shift it up or down one position:



In this example, we want to move MEMBER down, because it has dependencies on nearly all of the other tables. F_TRANSACTION has to go to the very last position because it depends on all of the other tables:

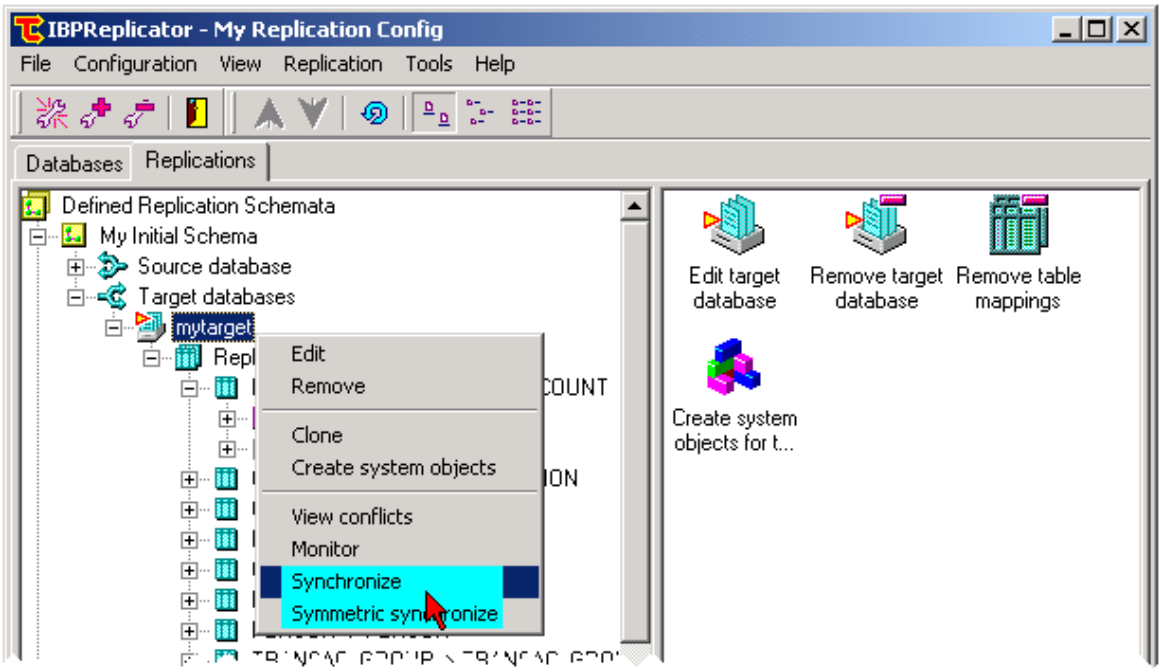
© 2007 IBPh en Ed rs



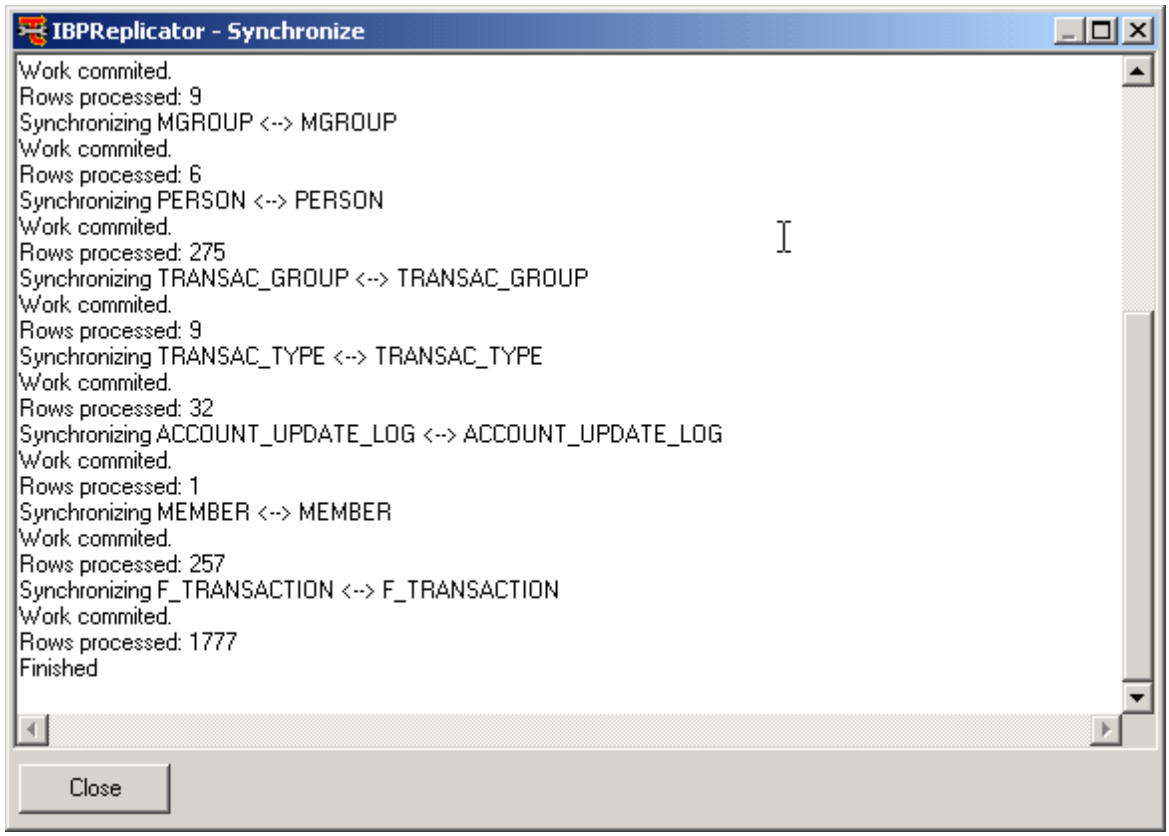
Simply continue to shift tables and procedures down and up until you are happy that you have taken care of the dependency issues.

Performing a Global Synchronization

Select the database node of the target you want to synchronize with the source, right-click and select the type of synchronization you want:



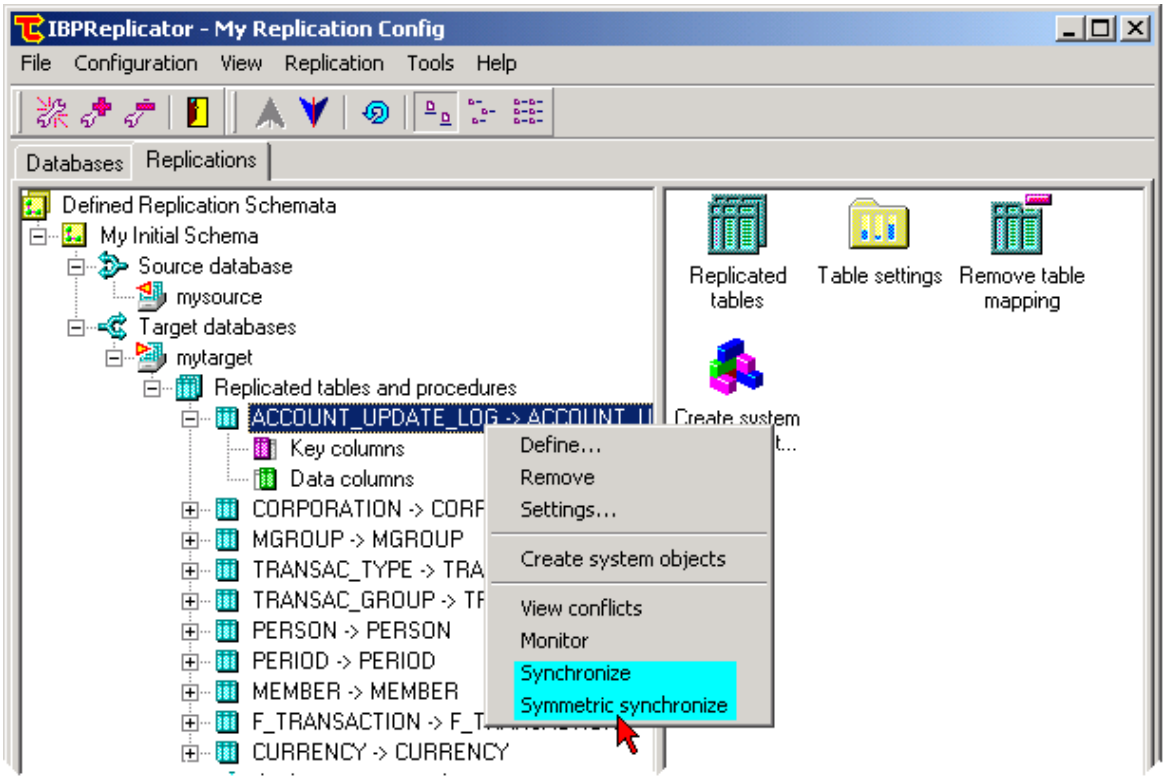
A monitor window will pop up and display the synchronization as it happens. The window will fill up very fast but you can scroll back to check what has been done:



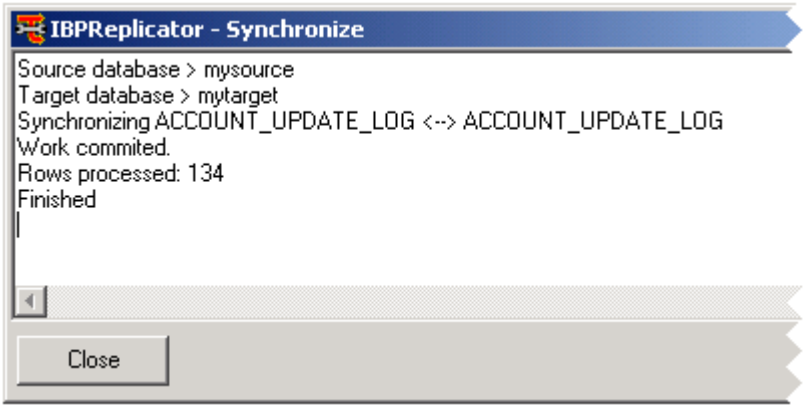
Single-table Synchronization

If you just want to synchronize an individual table, select the table in the Replicated Tables and Procedures node of the Replications tree, right-click and select a

synchronization option form the context menu:



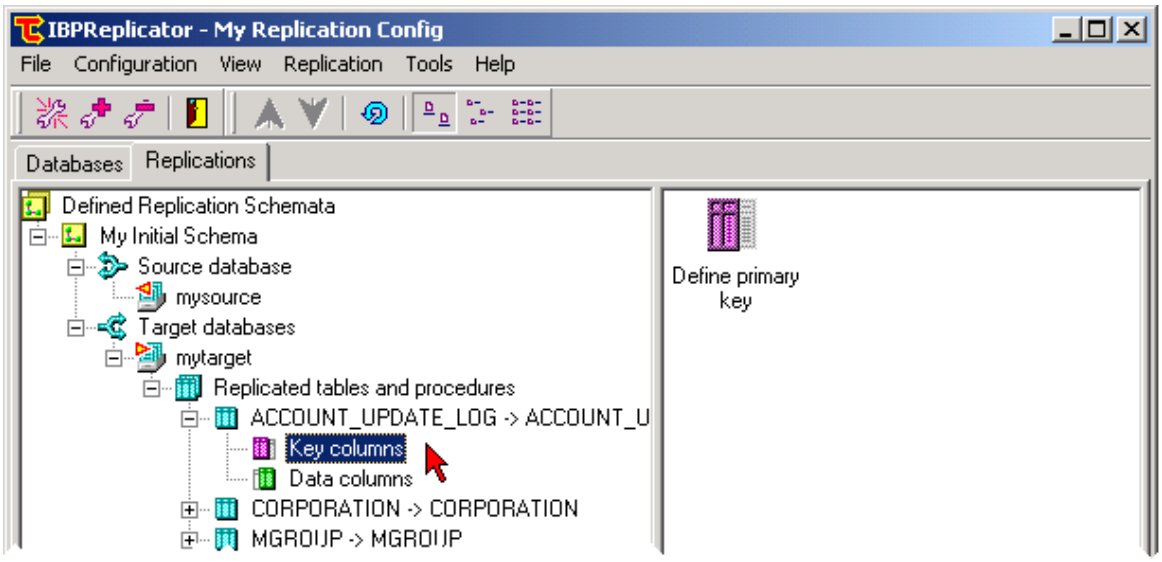
Select Synchronize to perform an asymmetric synchronization or (obviously) Symmetric synchronize if you want the full two-way synchronization. This time, the monitor is much more brief!



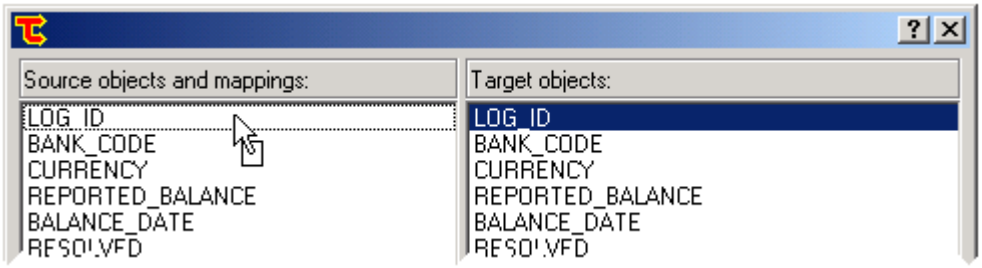
7.8 Identifying Primary Keys

Every table that is going to be replicated (from and to) must have a primary key. IBReplicator uses the primary key to identify rows on a target database for update and delete operations.

Expand the Target database tree node for the table pair for which you want to set up the primary key mapping and select the Key columns node:



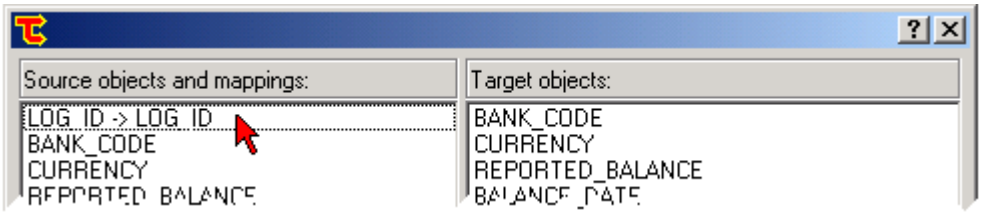
You can right-click and select Keys from the context menu or double-click on the Define primary key icon to open the Define key columns dialog box. You can also access this dialog by using the Replication | Columns | Keys menu option.



To map the source database and target database primary key columns click on and drag the target database primary key columns and map them to the corresponding columns in the source database.

NOTE Take care to ensure that all fields of a compound primary key are mapped. At the same time, be careful that you don't accidentally map any fields that are not elements of the key.

As you map the columns the mappings will appear in the Source objects and mappings pane:

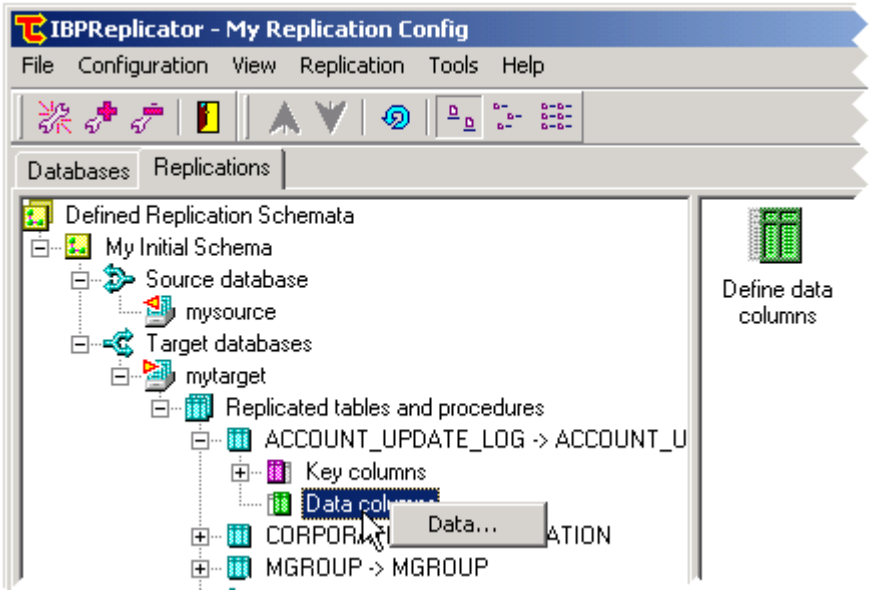


Once the mapping is complete, click OK close the dialog and store the mapping.

7.9 Choosing Replicated Columns

Once the primary key has been mapped, IBReplicator needs to know which columns need to be replicated.

Expand the Target database tree node for the table pair for which you want to set up the data field mappings and select the Data columns node:

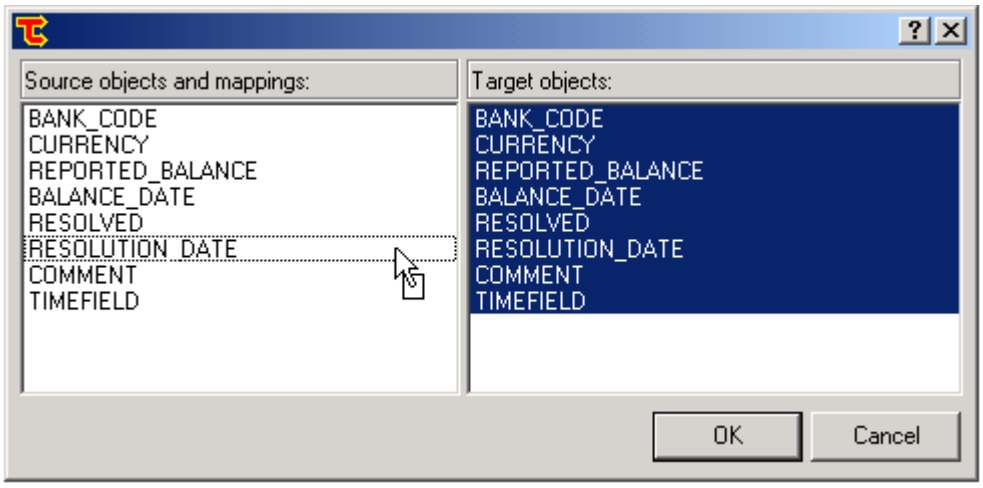


You can right-click and select Data.. from the context menu or double-click on the Define data columns icon to open the Define data columns dialog box. You can also access this dialog by using the Replication | Columns | Keys menu option.

The dialog looks exactly the same as the one you used to define the primary key columns. As before, map the target table columns to the source columns by selecting and dragging a target column name to the corresponding column name in the source panel.

> TIP You can use the standard Windows ctrl-click and shift-click to multi-select column names. When you drag a group, make sure the drag cursor is focused on one of the source column names when you release the mouse button.

When you are finished, the result might look similar to this:



Click OK when you have done with this table and continue with your other tables.

NOTE If you are replicating only keys and no other columns then there will be nothing to do here.

7.10 Creating Source System Objects

The final step, once the rest of the replication schema has been defined, is to ask the Replication Manager to create the tables, triggers, etc., in the source database that will be used for the replication operations. These structures are referred to as Source System Objects.

It is recommended that you perform Create system objects when you are the exclusive user on the source database, particularly if the source database is on a Firebird Classic server. The triggers created by IBReplicator will be seen by other database users when the database metadata cache is refreshed. On Classic, each client has its own cache and won't see the changes until next time that user logs in. It is less of an issue with Superserver, since the cache is shared by all clients and none will see the changes until after all users have logged out.

As usual, you have several ways to make this request. This time, you will be operating on the Source database, so select it in the Replication tree and do one of the following:

- right click on the Source database node and select Create System Objects from the context menu
- double-click on the Create system objects icon in the right pane
- choose Replication | Source | Create system objects from the menu

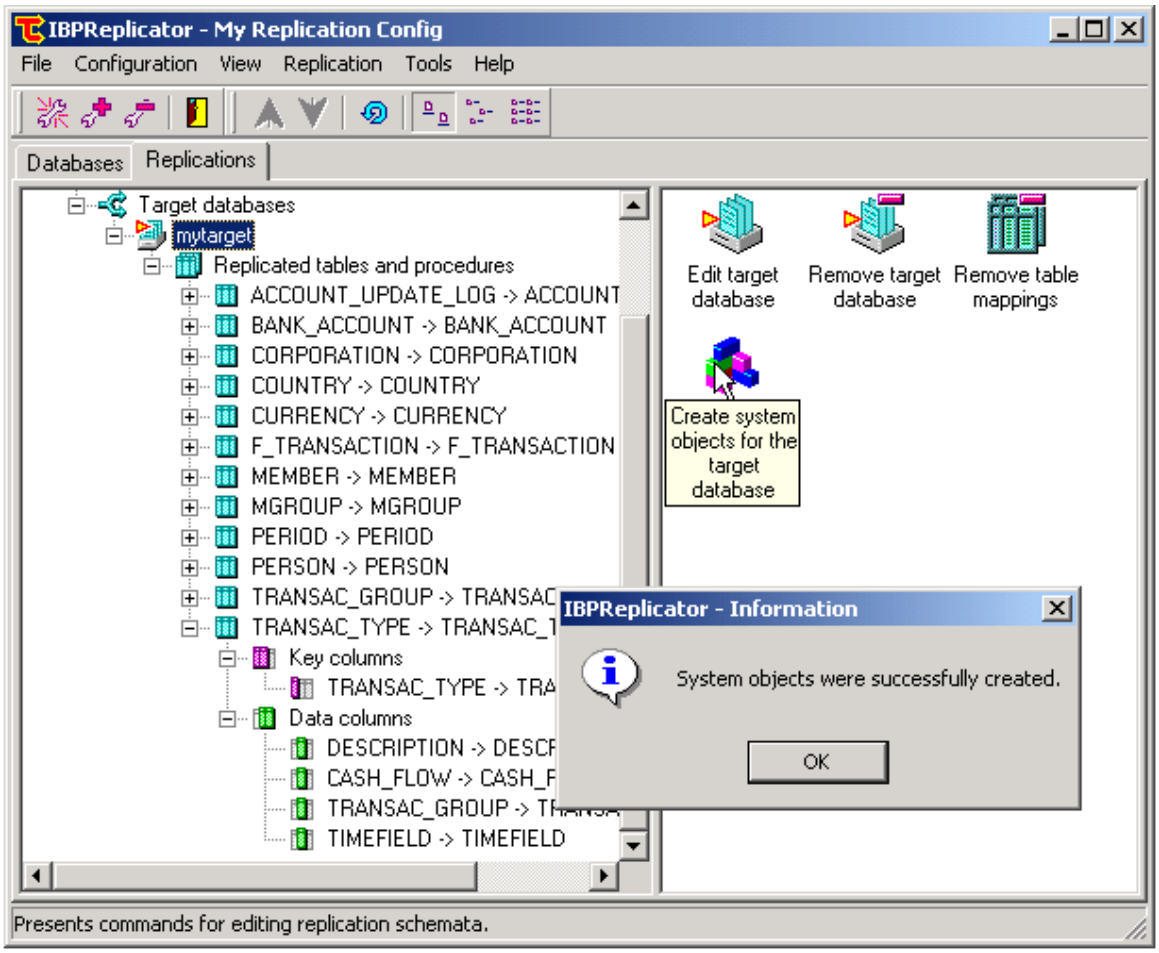
For convenience, you can access Create system objects from the target database node to avoid hopping about the tree—the objects are always created in the source database, regardless.

TIP You can create all of the system objects globally by using the main menu option.

If there are any problems--for example, missing primary key assignments--you will see

an error message and no system objects will be created. Correct the situation and try again.

If all goes well, you should see the success message:



Removing System Objects

Select the source database in the Replication tree and double-click on the Remove system objects icon to remove all the IBReplicator defined system objects from a source database.

Chapter 8 - Managing Replication

Chapter 8 - Managing Replication

The topics in this chapter concern the actual running of your replication services and the tools and utilities provided to do so.

On Windows, you will (ideally) run IBReplicator as a service. On Windows, the installer will set everything up to run the service according to your preferences.

For running IBReplicator as an application, a graphical interface is available from the Start Menu to set up how you want it to run. Some control aspects can also be run from the command shell 79, either directly or in a .bat file.

If you need to control IBReplicator wholly in Linux, only a command-line interface is available currently for controlling the services.

8.1 Windows Operation

The Windows Executable

The executable that provides the replication services on Windows is ReplServer.exe.

Previous versions of the IBReplicator software provided a graphical console for managing the service and the associated scheduler service. In v.2.5, the scheduler is incorporated in the server program, while the replication service itself will run according to the choices you made during installation.

Essentials Before the Replication Server Will Run

Make sure that

- your configuration is in a sufficiently complete state that replication can occur
- your server and replicant licences are installed and that you have enough replicants to perform the configured replication[s]
- the source and target databases have the necessary permissions granted for the REPL user or, if you are using a role (e.g. REPL_ADMIN as in our examples) that the role has been created and the permissions granted. (And remember to grant the role to REPL!)
- you have set the default configuration database—otherwise it won't be found and the server program will refuse to start!
- the Firebird or InterBase servers are running on the source[s] and target[s]

Running IBReplicator

The replication server program can run on Windows as a service or as an application.

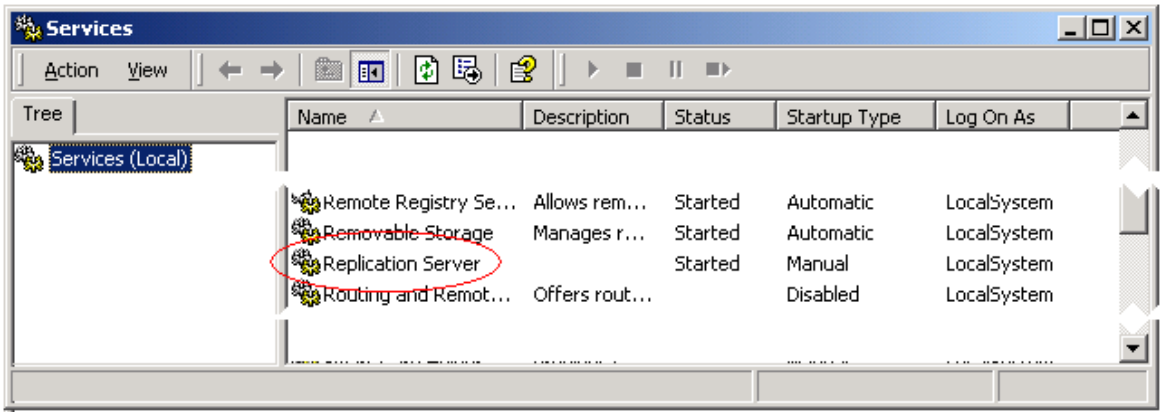
© 2007 IBPh en Ed rs

As a Service

As a rule, on the Windows service-capable platforms—Windows NT, 2000, XP Professional and the 200X Server versions—you will be running replication as a service.

Starting the Service

The Windows installer should have installed the service for you. You can check this by inspecting the Services display by way of the administration tools on your server:



- If you elected to have the service start automatically, it will have the Startup Type "Automatic" and should be running ("Started") unless someone has stopped it.

- If you elected to have it start manually, or it has been stopped (shut down), you can start it the same way as you would start any other service, i.e. by right-clicking on its entry in the list and selecting "Start" or "Restart", or by selecting its property sheet and manipulating it from there.

As an Application

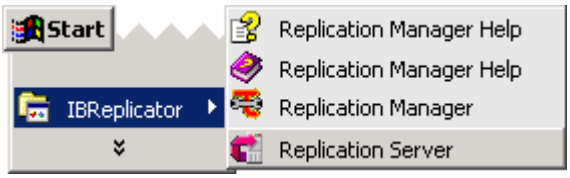
Resources permitting, it is possible to run ReplServer.exe as a stand-alone application. It can be started from the supplied graphical interface or from the command-line (including a batch file).

- On Windows 9x and ME you have no alternative, since these platforms do not support services.

- On the other platforms, you can run it as an application to perform occasional replication runs, that can use an alternative configuration database, if necessary.

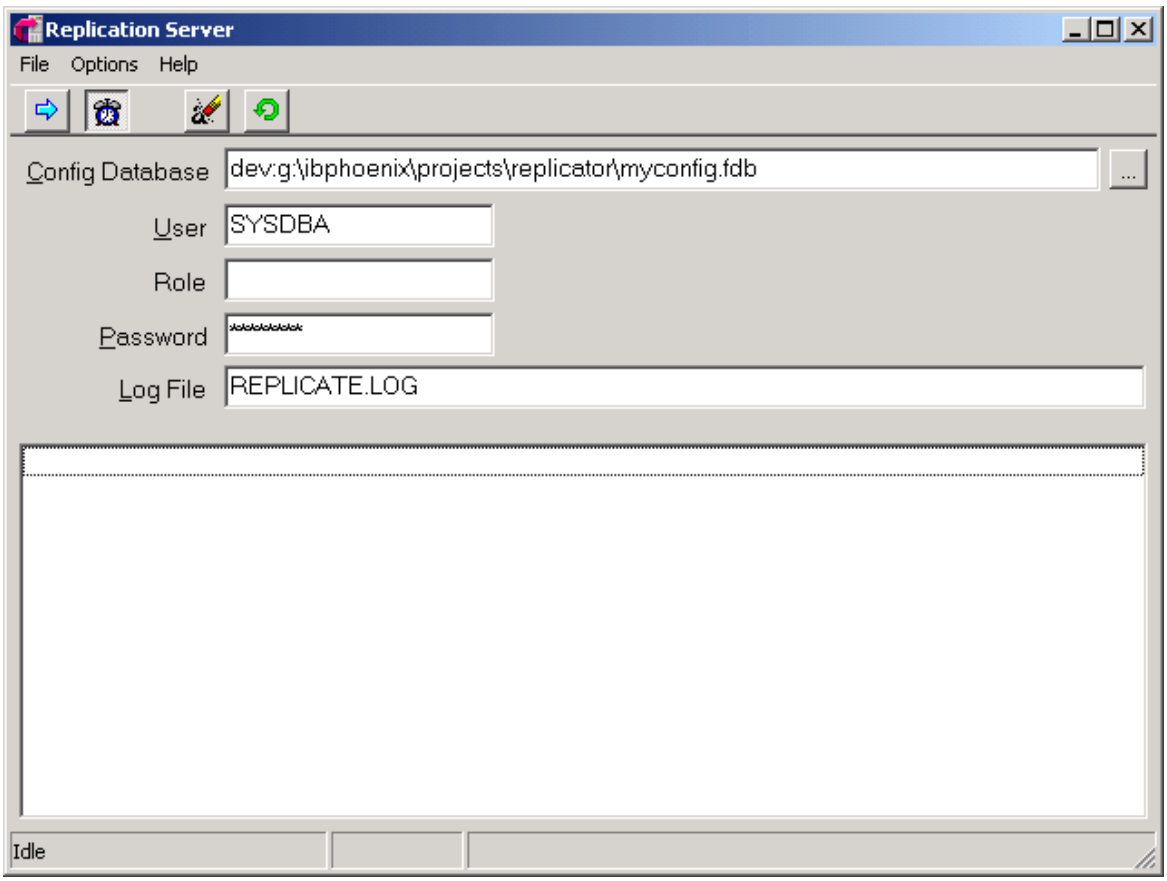
To start ReplServer.exe as an application from the graphical interface, click its icon in the Start Menu:

© 2007 IBPh en Ed rs



The Application Interface

The graphical interface appears, populated with the parameters for the default replication that it has started:



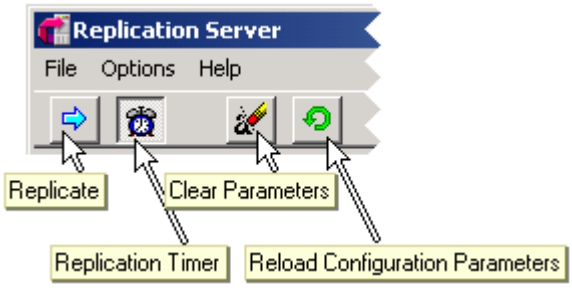
If you watch long enough, the log window will fill up with an account of the replication that is going on. If it is doing what you want, just minimise the display and let 'er rip. You can maximise it and look in on your replication at your pleasure:

```
11/05/06 23:08:59: ******************** Start Replication ******************
Replicate My Initial Schema, mysource -> mytarget
11/05/06 23:08:59: Connected to DB: dev:g:\IBPhoenix\projects\data\sourcedb
11/05/06 23:08:59: Connected to DB: shorty:d:\projects\data\targetdb.fdb
11/05/06 23:08:59: Disconnected from DB: dev:g:\IBPhoenix\projects\data\sou
11/05/06 23:08:59: Disconnected from DB: shorty:d:\projects\data\targetdb.f
TOTAL STATISTICS:
0 Hours, 0 Minutes, 1 Seconds
Inserts  :         3
Updates  :         7
Deletes  :         2
Conflicts:         0
Errors   :         0
11/05/06 23:09:00: ****************** Replication Finished ******************
```

If your purpose is to do more than just sightseeing, use the input fields and the toolbar buttons to set up and run the application using different parameters.

The Toolbar Buttons

You can use the toolbar buttons to load a different set of parameters and run a different replication—or to set up and



In the example, the Timer button is depressed, indicating that the configuration is doing (or will do) a timed replication. If you are going to do a different replication as a one-off, you would click this button off.

Click the Clear Parameters button. Note that the application will start to nag at you to enter something if you wait around too long! Proceed to make your alternative entries:

- In the Config Database field, browse for (or type in) the full connection path to an alternative configuration database
- Fill in the User and Password fields and, if needed, the Role
- Change the default log to a different valid path and filename if you wish

Click the Reload button to load the new parameters.

Click the Replicate button to start replicating (or to do a one-off replication, if the Timer button is off).

© 2007 IBPh en Ed rs

Managing the Server

Once the server is running, you can stop and restart it. How you do so depends on whether it is running as an application or a service.

To stop the server when it is running as a service

Either:

- Open the Replication Manager and execute (from the Menu) Replication|Tools|Notify Server|Shutdown Server or press Ctrl-U; or
- Stop the service from the Services applet in the Administration Tools

Note also that you can stop and restart the service from the Services applet by choosing Restart.

To stop the server when it is running as an application

Simply close (Exit) the application. You can do this either

- from the menu of the graphical interface program; or
- by selecting it in the Windows Task Manager list and clicking End Task

8.1.1 Command-Line Operation

The server program on Windows (ReplServer.exe) can be run from the command line using option switches.

> Note that the syntax and semantics of the switches are not the same as those used for command-line operation on Linux.

Program Switches for ReplServer.exe

Switch and Argument	Effect	Comment
--application or -a	Run the server as an application	The default is for it to run as a service—recommended if available. However, you can run a non-default replication as an application whenever needed. It is probably a good idea to avoid running two replications simultaneously, though, in case it causes lock conflicts.

© 2007 IBPh en Ed rs

--min or -m	Run (with -a) minimised	Minimises the shell window to the Task Bar
--go	Start replication automatically and quit after replication is complete	Convenient for running replication from another application/script
--log_to_file=n	n= 1: Errors Only 2: Warnings and Detailed Error Description 3: Information 4: Detailed Information 5: Debug 6 & 7: Thorough debug, applicable to debug builds only	Default is 0 (no logging). If a support consultant asks you for a "debug log", level 5 is the one to generate. Higher logging levels apply to a debug build, which is not part of the normal software distribution.
--log_to_window=n	Values for n as above. Echoes the log to the command shell	
-c=<cfg>	<cfg> is the full connection path to a specified configuration database	Can be used to run an alternatively configured replication
-p=<password>	<password> may be needed if it the config database specified is not the default one and the access password is different to that of the default config DB	
-u=<username>	<username> if needed for the situation described above	
-r=<role>	<role> if needed for the situation described above	
--pd	Period disconnect/reconnect	Disconnect from the configuration database between replications. The default is to stay connected.

Installation Switches

In case you need to know the installation switches for making your own service installer script, they are as follows:

© 2007 IBPh en Ed rs

/install	Install the replicator service
/uninstall	Uninstall the replicator service. (This uninstalls the service, not the software!)
/auto	Have the service start automatically
/dep=<service name>	Used for passing the <service name> of a service that the replication service name depends on

8.2 Linux Operation

The Linux Executables

Two executables are provided for operating a Linux server as an IBReplicator server:

- replserver is the server
- replmgr is a utility for starting and stopping the server

Replication Manager on Linux

On Linux, the Replication Manager is currently a command-line utility only, for controlling one or more replication processes that you have previously configured via a Windows client.

> NOTE IF you do not have a Windows machine available for defining your schemas and installing licences, it is possible to run the Windows replmgr.exe program in wine, the Linux Windows emulation system.

The command replmgr takes switches, with or without obligatory parameters (arguments), according to the way you want a replication process to run (or to stop running!). Some switches are required, others are optional.

Switch Format

> Note that the syntax and semantics of the switches are not the same as those used for command-line operation on Windows.

The format for supplying switches and their arguments is

```
-x=parameter
```

where -x is a hyphen followed by a character representing the switch and parameter is the actual value for the argument.

Required Switches

The replmgr command must include these switches:

Switch and Argument	Parameter	Comment
-u=username	Should be the Firebird or InterBase user name of the user that is connecting to the configuration database.	Make sure this user exists in the security database on the server where the configuration database resides. Not case-sensitive.
-p=password	The password of this user.	Passwords are case-sensitive.
-c=configuration database	Valid path and filename for the configuration database that is to be used for this task.	Because events are used to communicate with a running replication server, it is essential to use the same configuration database parameter when performing each action regarding a specific instance of the replication server. Remember that filesystem names on Linux are case-sensitive.

Optional Switches

These switches are optional:

Switch and Argument	Parameter	Comment
-a=action start|stop|force	Use only one parameter.	Both stop and force can be used stop the replicator process. With stop, a running replication will complete first; with force, a running replication will be interrupted.
-m=mode window|daemon	Use only one parameter.	In window mode, the replication process runs in a shell and can optionally display the replication log, while in daemon mode it runs in background, on demand, and log only to a file.
-r=role	Valid role identifier of a role (package of privileges) optionally defined in the configuration database for the user doing the replication.	The environment variable ISC_ROLE, if defined, will be used as the default role if the -r switch is not used; otherwise the role is treated as a no-op (ROLE NONE on Firebird).
-d=debug	Run in debug mode.	This is logging level 7, the most verbose level. If a support consultant asks you for a "debug log", this is the one to generate.
-log_to_window= logginglevel	1: Errors Only 2: Warnings and Detailed Error Description 3: Information 4: Detailed Information 5: Debug	As the logging switches are optional, the default logging level is the one defined in the configuration. Debug level (5) can be useful when you need to troubleshoot a configuration.
-log_to_file= logginglevel	1 to 5, as above.	Same as above, writing to the log file. You can log to window and log to file simultaneously if you wish.
-help	No parameters	In window mode, displays a tight summary of these switches and parameters.

Running replserver Independently

It is possible to run one or more instances of replserver independently of the controlling replmgr process, i.e. as a "stand-alone" process. All of the same switches are available except the -a (action) switch. Stopping a stand-alone replserver instance requires an explicit kill command.

© 2007 IBPh en Ed rs

Multiple Concurrent Replication Processes

Multiple replication processes can run concurrently on the same host system, provided each instance of the replication server uses a different configuration database.

> Trying to run multiple instances of replmgr/replserver against the same configuration database will have unpredictable results.

If you are running multiple instances of the Replication Server on the same system (each connecting to its own configuration database), you will need to use replmgr -action stop multiple times to stop each server, in each case using the same configuration database path that you used to start that process.

Environment Variables

Four environment variables can be useful when used in conjunction with IBReplicator, to default the values of the corresponding arguments:

- ISC_USERNAME - Contains the required username
- ISC_PASSWORD - Contains the required password
- ISC_ROLENAME - Contains the required role
- IBPATH - Contains the name of the directory where the Firebird or InterBase server is found

> CAVEAT Be careful about relying on any of these environment variables if you are running multiple concurrent instances of replmgr or replserver.

Examples

```
./replmgr -u =SYSDBA -p=masterkey -a=start -m=window -c=config.fdb
```
Starts the server and logs to a window

```
./replmgr -u=SYSDBA -p=masterkey -a=start -m=daemon -c=config.fdb
```
Starts the server and logs to a defined log file

```
./replmgr -u=SYSDBA -p=masterkey -a=stop -c=config.fdb
```
Stops the server

8.3 Management Tools

Replication Monitor 85

Replication Scheduler 86

Schema View 94

Notify Server 88

View Conflicts 89

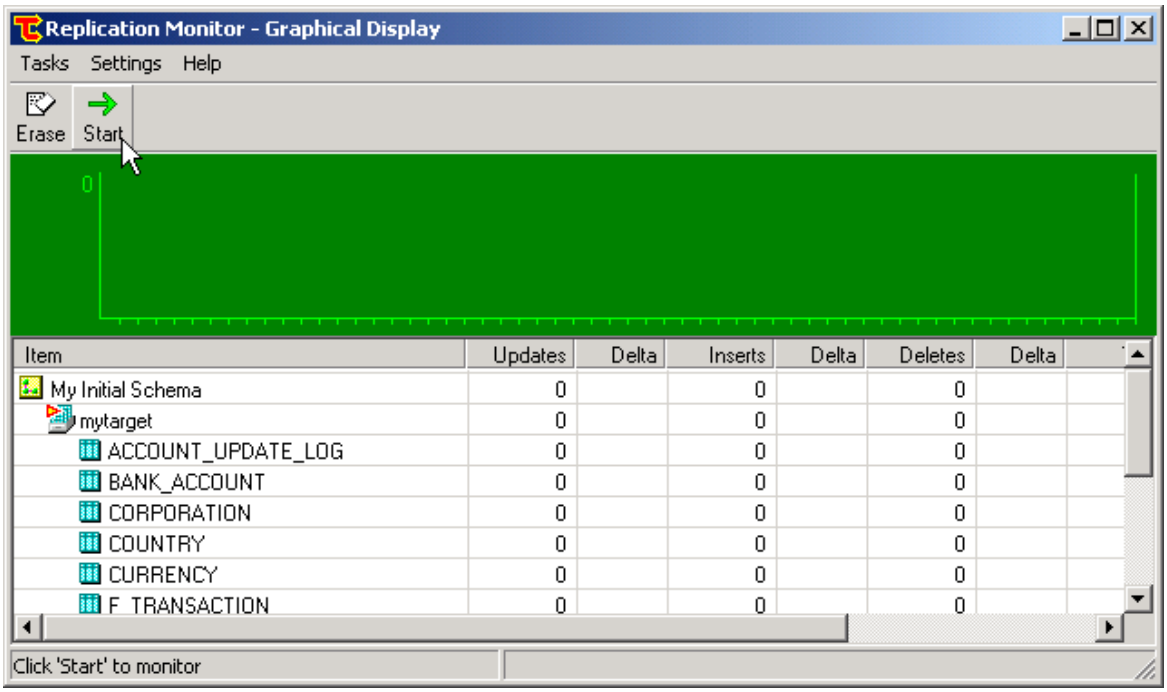
© 2007 IBPh en Ed rs

License Manager 91

8.3.1 Replication Monitor

The Replication monitor displays a graph of replication activity for the replication schema that is currently selected in its tree.

Open the monitor by selecting Tools | Monitor from the menu:



Activating and Stopping the Monitor

Click the Start button to begin monitoring the selected types. While monitoring is going on, the button caption changes to Stop. Click this button again to stop the graphing process.

Keeping Things Rolling Along

You can graph just one type of replication statistic or any combination of types. While the monitoring and graphing is going on, you can

- clear the display by clicking the Erase button
- select schemata in the tree at any time
- set error and conflict notification by selecting Settings | Error/Conflict Notification. A dialog appears, enabling you to set options such as display a message or mail the designated person when an error or conflict occurs

© 2007 IBPh en Ed rs

8.3.2 Replication Scheduler

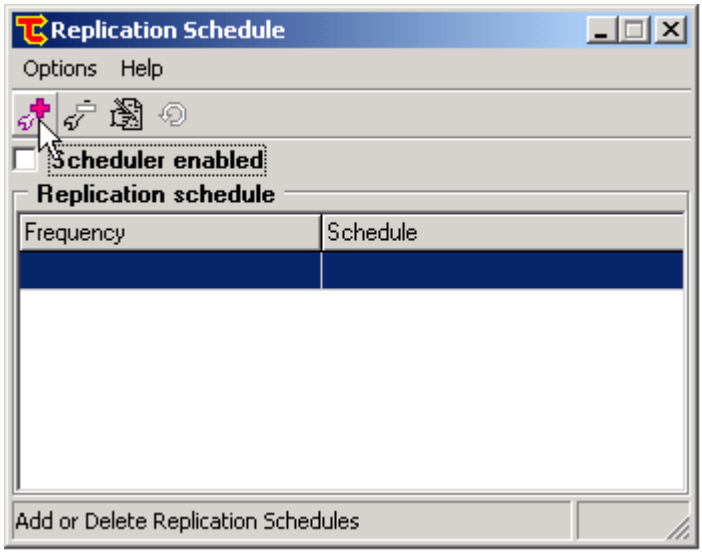
The Scheduler provides the ability to manage replication at custom periods by setting a predefined interval (in minutes) or by frequency—Once Only, Periodically, Hourly, Daily, Week days, Weekly or Monthly.

IBReplicator 2.5

In version 2.5 of IBReplicator, the scheduler is merged with the Replication Server and this tool becomes the only method of defining when a replication takes place.

Setting up a Schedule

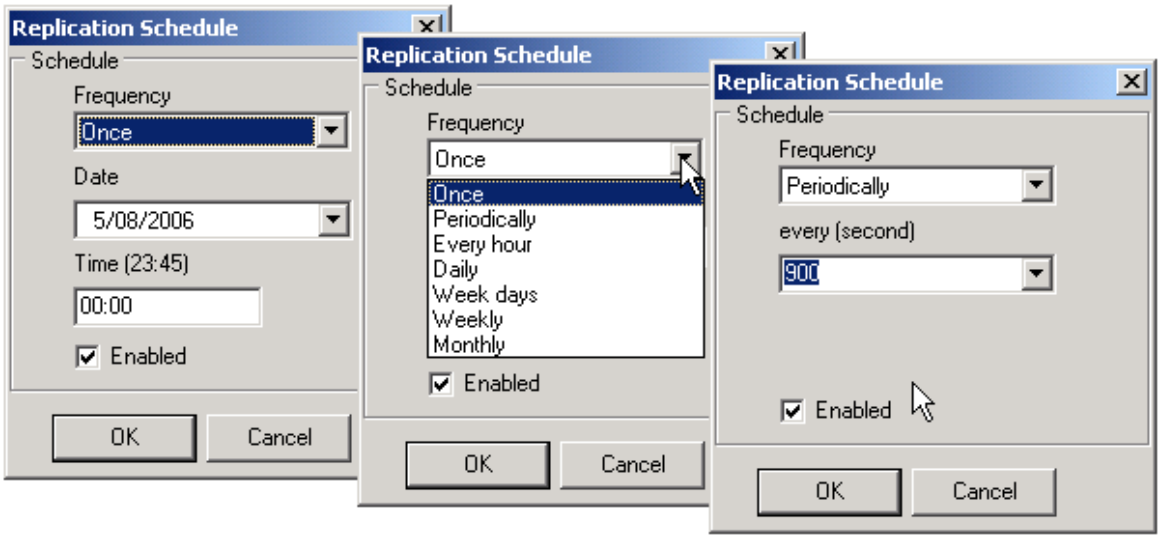
To set up a replication schedule, select Tools | Scheduler:



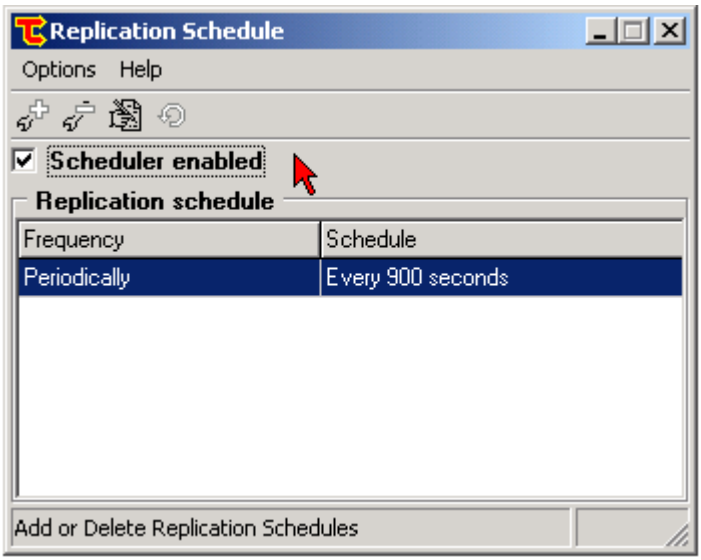
Adding a Schedule

To add a schedule, either

- drop down the Options menu and select Add, or
- click the Add button on the toolbar, or
- simply press the Ins[ert] key on your keyboard

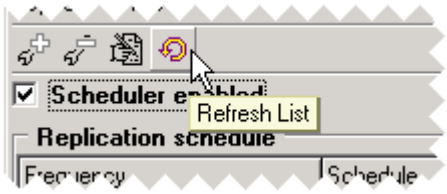


Initially, the add dialog shows the default "schedule", which is to replicate once, at midnight on the current day—which won't happen today, of course! In this example, we are choosing a periodic replication every 15 minutes. Of course, you can drop down the options and pick the ones that you need. Click OK to add the schedule.



Starting the Scheduler

- Check Scheduler enabled
- If you want to hear an audible signal when a replication starts, also check Beep when Replicate
- To update the Scheduler and replace any schedule that is currently in effect, select Refresh List from the Options menu, press F5, or click the Refresh List button on the toolbar:



Altering the Scheduling

Schedules can be added or deleted as required. Always remember to Refresh List if an existing schedule is in effect.

TIP If you need only occasional replications, you need not schedule replications at all: simply use the menu option Tools | Notify Server 88 | Replicate Now to replicate whenever it suits you.

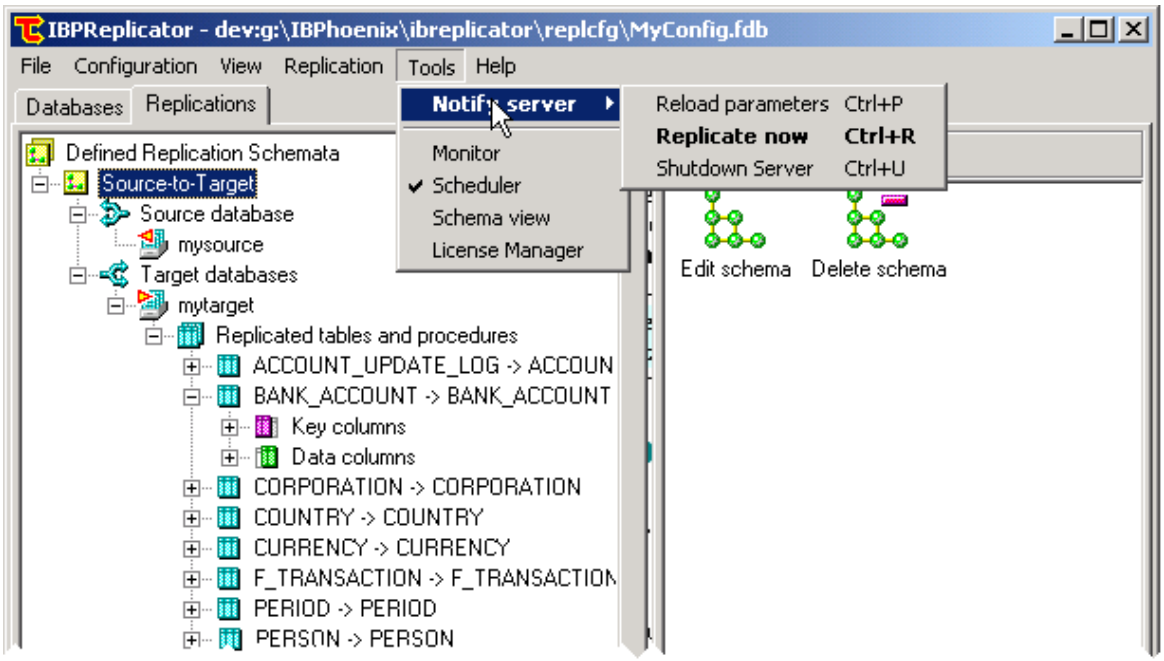
8.3.3 Notify Server

It is possible to intervene in the running Replication Server's activity and instruct it to perform certain actions. You can co this intervention via the Notify Server utility, from the main menu of the Replication Manager.

Because this utility uses events to alert the server, it is essential to open ports in your firewall on both the server and the target database hosts for the event traffic.

The Notify Server Options

Tools | Notify Server offers three options:



Reload Parameters (Ctrl-P)

Instructs the server to reload all its settings from the current configuration databases, usually because the configuration has been altered and there is no need to shut the server down and restart it.

Replicate now (Ctrl-R)

Instructs the server to replicate immediately, without waiting for its next scheduled replication. If you need only occasional replications, you need not schedule replications at all: simply use this command to replicate whenever you like.

Shutdown server (Ctrl-U)

Instructs the Replication Server to shut down immediately.

8.3.4 Conflict Resolution

When something goes wrong because of a conflict between the source data and the target data, you want to know about and fix it.

If you are checking the log, you will be aware of any conflicts encountered, because they are summarised there after each replication. There are other places in Replication Manager where conflict information can be viewed in more detail.

Viewing Conflicts

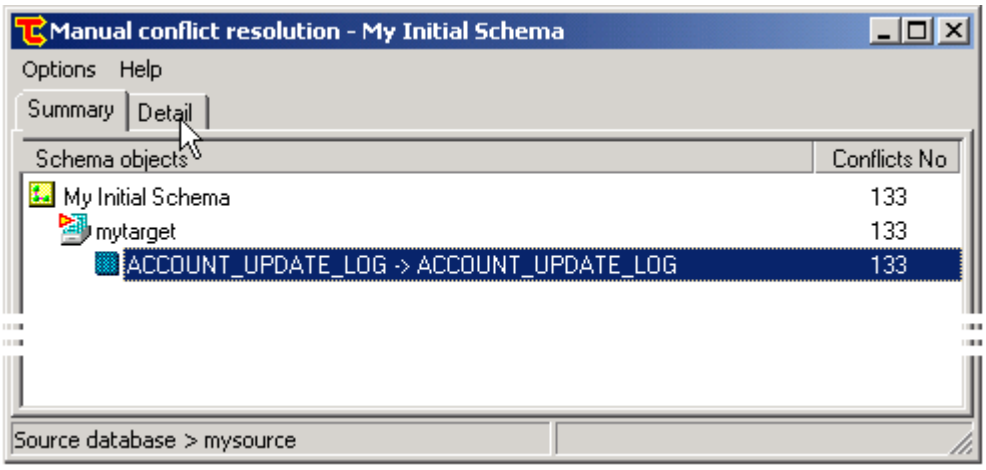
With the replication schema you are interested in selected, either

- click Replication | Tools | View Conflicts; or
- right-click on the schema node of a defined source or target database in the Replication tree, or anywhere else within that schema branch, and choose View conflicts from the context menu

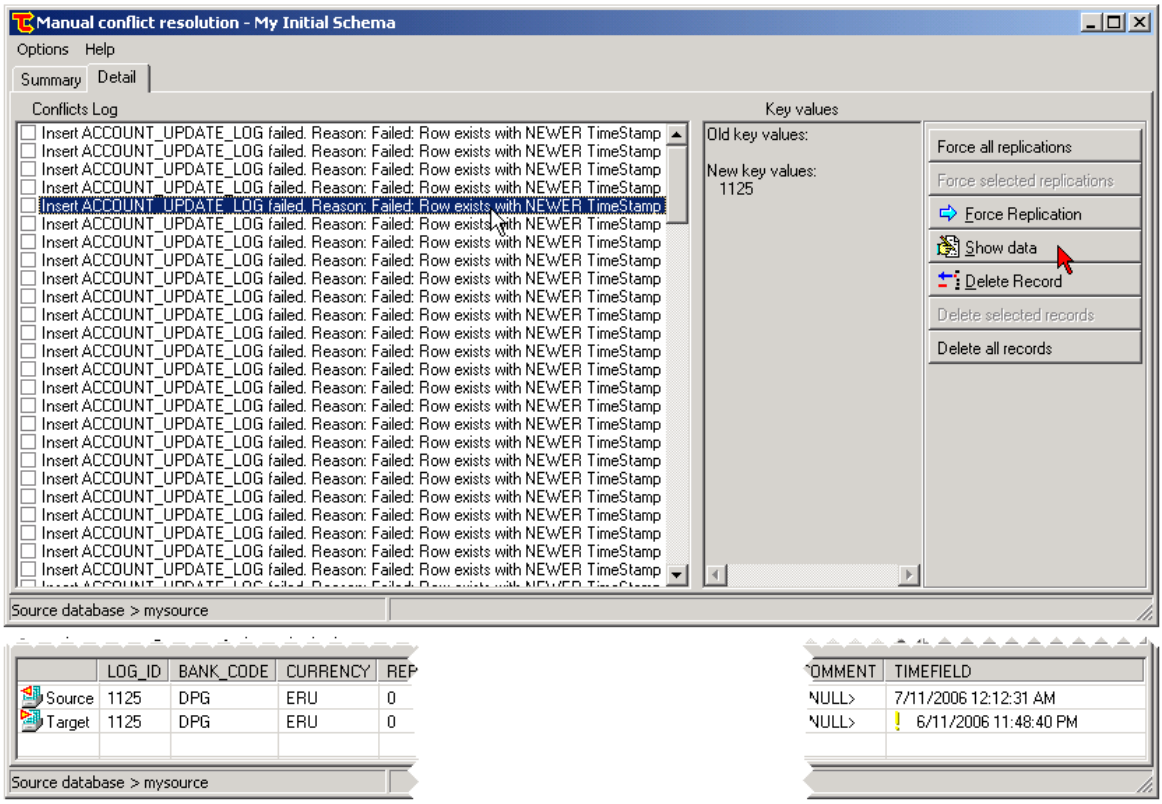
NOTE to focus on all conflicts that may have occurred when replicating from a particular source database or to a particular target database, have that database node selected.

If there are no conflicts, the conflict viewer simply says so. If there are unresolved conflicts, the viewer displays them:

© 2007 IBPh en Ed rs



Here we see that the target database had 133 conflicts. We want to open the Detail tab to see what those conflicts were:



Every conflict for this table is the same—so we select a record and click the Show Data button at the right to find out the offending field. A new panel opens beneath the Detail display and shows the conflict for this record. Ah—yes! we previously used a synchronization to get all the records from the source table over to this table, but the server clock at the target was 20 minutes fast!

Resolution

If any conflicts are listed, you can click on one to highlight it and select a button from the Options menu to resolve it:

Force All Replications	IBReplicator will try to replicate all the problem rows during its next scheduled replication
Force Selected Replications	If you want to force replication on some but not all of the conflicting records, use the checkboxes at the left of the detail display to select/deselect. Then, the button will become enabled and you can use it to tell IBReplicator to try to replicate those rows during its next scheduled replication
Force Replication	IBReplicator will try to replicate the problem row during its next scheduled replication
Delete record	Delete the selected conflict. Data in source and target tables will stay as they are
Delete selected records	If you want to delete groups of conflict records, select those you want deleted. This will enable the button, ready for you to click it to have the conflict records abandoned.
Delete all records	Delete all the conflicts. Data in source and target tables will stay as they are

After an option is executed, the record in the manual log will be deleted and the conflict will disappear from the display, considered to have been resolved.

Finally, the view reports the "all clear" and you can close it.

8.3.5 License Manager

About Licensing

Evaluation Licence

No specific restrictions are placed on the distribution of the Evaluation version, except for the following:

- Once entered, the evaluation license lasts for a period of 14 days
- The evaluation licence is limited to one server, with its embedded replicant licence, and three more replicant licences

Normal Licensing

On each host machine that acts as a replication server, one Server (or Server Lite) license must be installed. Each replicated database—whether source or target—needs a replicant license. The server license (or lite license) comes with one replicant license included.

In the simplest scenario, you will need to buy at least two licences:

1. a server licence (that automatically includes one replicant)
2. one other replicant licence

So, for example, a machine that acts as the replication server and has a single Source database, and which is replicating to a remote Target database, will need only two

© 2007 IBPh en Ed rs

licenses bought: a server license and a replicant license.

Same database used as multiple replicants

No database needs to be licensed more than once. If the same Source or Target database is being controlled by more than one instance of the replication server, then only one replicant needs to be bought for that database. This license will be installed in all configuration databases involved.

Server vs Server Lite

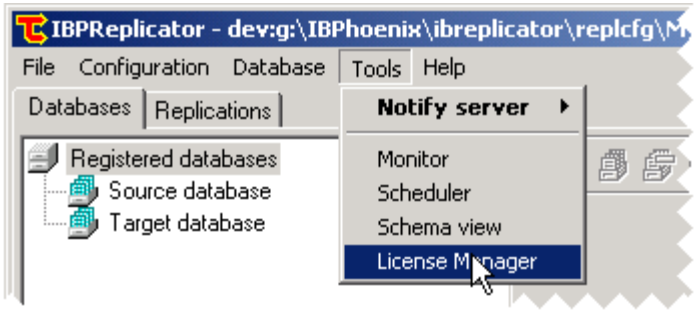
The distinction between the Server and the Server Lite licenses is that the Lite license was created primarily for laptops needing a Replication Server software installation for synchronising with an organisation's main database.

The Lite license is substantially cheaper because it is assumed that laptops are stand-alone systems, not acting as Firebird or InterBase servers for remote clients.

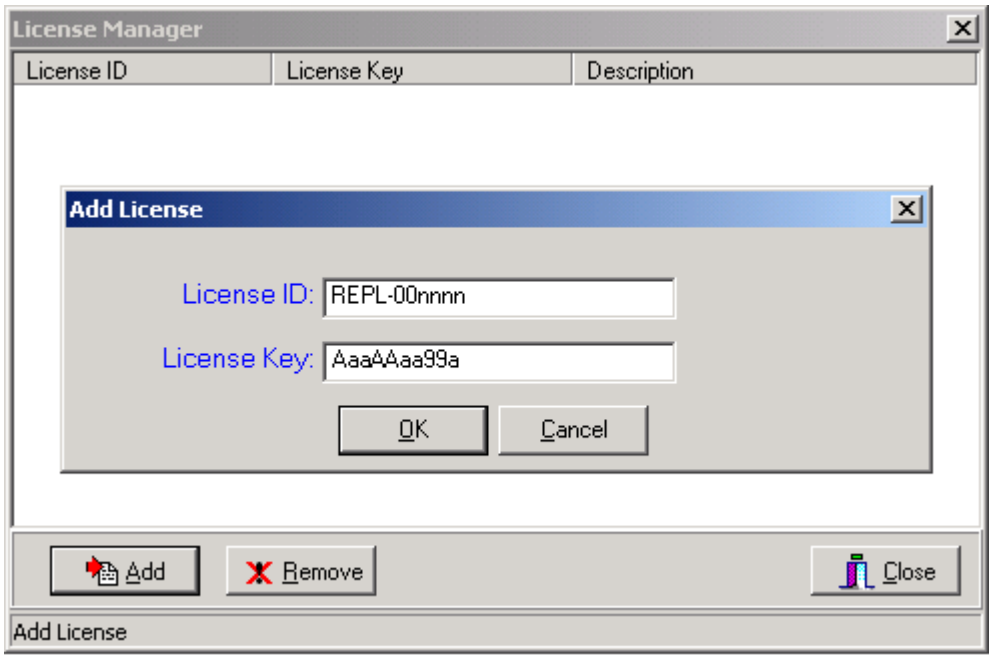
The normal Server license is intended for multi-user servers, which may be replicating to and from multiple remote servers.

Installing Licences

Once you have created your first configuration database, the Tools menu of the Replication Manager becomes visible. It contains the License Manager applet for installing your licences:



Click the License Manager item to start the applet, then click the Add button to add your first licence.

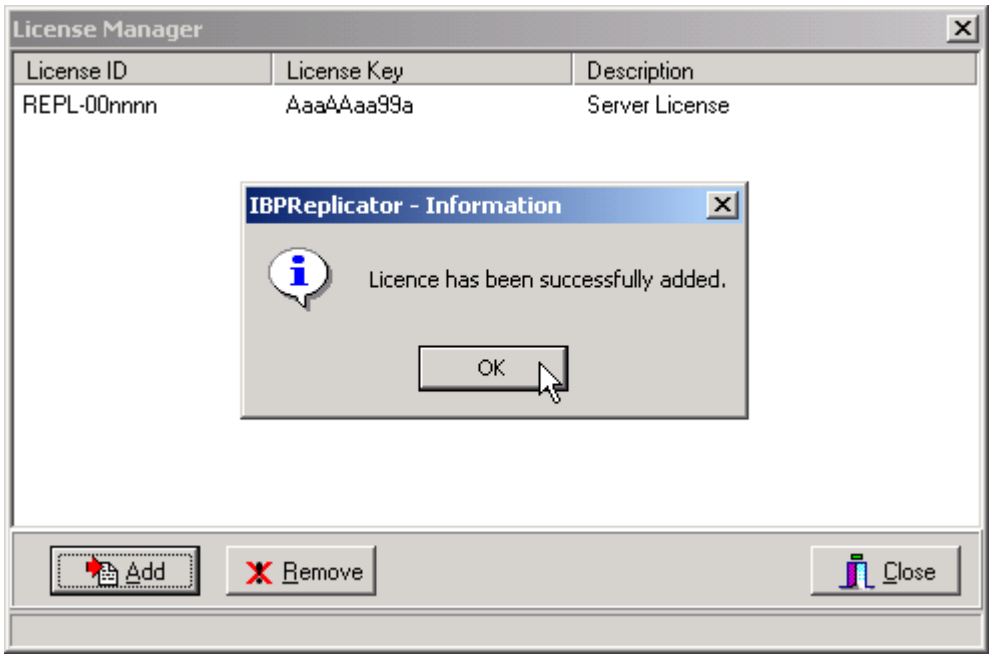


Evaluation License

Enter EVAL as the License ID and leave the License Key field to be filled automatically.

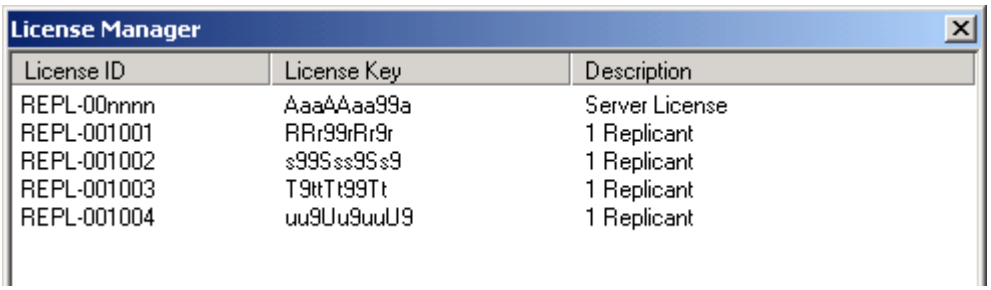
Normal Licensing

Type in or copy/paste the first of the License ID and License Key value pairs from your licensing document and press OK. The first licence registration appears in the display with a "success" message:



If you get an error message, try again. The codes are case-sensitive.

Continue to add each of your licences in the same fashion, until you have registered all of them. They should appear on the display as a list:



Don't worry about the order in which the licences are registered. When the time comes for action, each server is only concerned that there are enough available licences of each type to perform the tasks it is asked to do.

When you are finished, click the Close button to terminate the applet. You can access the License Manager any time by returning to the Tools menu or running it from the Start Menu.

Installing Licences for Linux

In the event that you have created or restored your configuration database on a Linux server and you don't have the Windows Replication Manager connected, you can easily install your licences using ISQL or another Linux-based DSQL tool.

You should log in to your configuration database using the username and password of either SYSDBA or the user that you specified as the replication administrator.

The statement is a simple insert. If you have multiple licences to install and you are using a tool that can handle input parameters (such as FlameRobin), you can define a single, parameterised statement, prepare it and then enter each License ID and License Key into the prepared parameters for as many executions as it takes:

```
INSERT INTO LICENSE (LICENSEID, LICENSEKEY)
VALUES (:LICENSEID, :LICENSEKEY)
```

If you are using ISQL, you will need to hard-code the VARCHAR values, perhaps in an INPUT script:

```
INSERT INTO LICENSE (LICENSEID, LICENSEKEY)
VALUES ('REPL-0nnnn', 'AaaAAaa99a');
```

8.3.6 Schema View

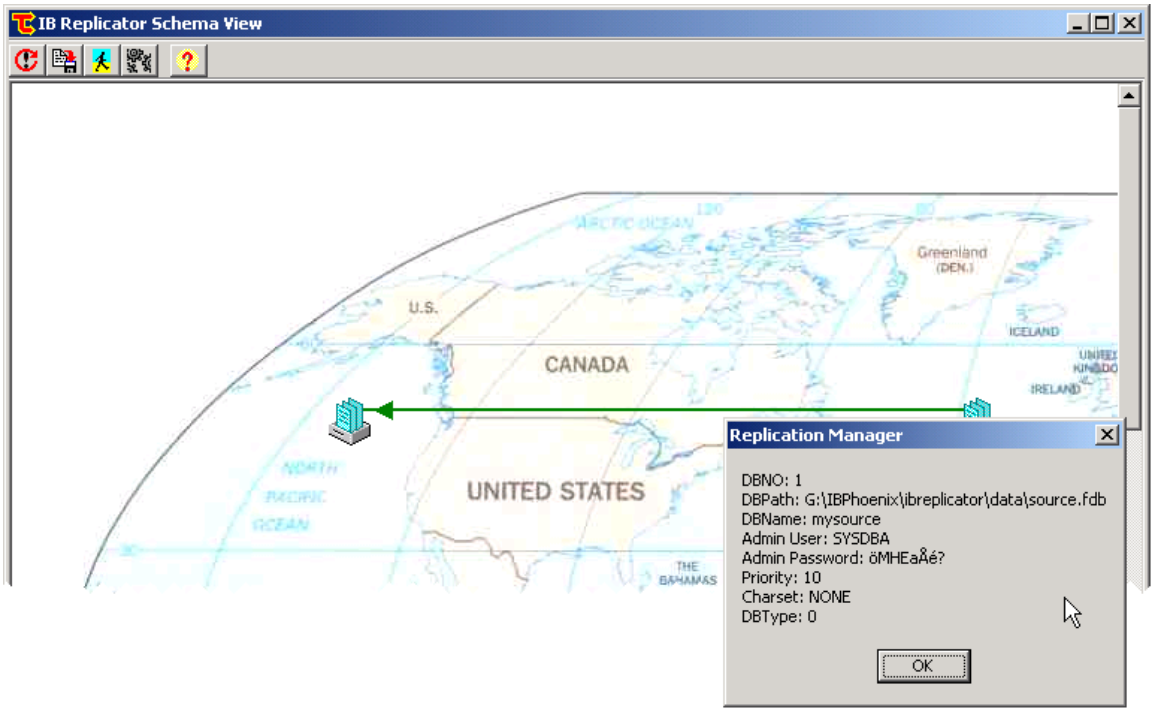
Schema View, accessed from the Tools menu, provides a way of storing a visual "map" of the physical layout of your replication servers.

Although this tool is aware of how the sources and targets in a configuration relate to one another, at this point it is not a visual doorway to your replication schemas. However, it can be useful as a quick way to identify information about the databases in your replication system and to store it diagrammatically.

© 2007 IBPh en Ed rs

Right-click on a database icon:

Click on the tag to see the database information:



You can pick up a database icon and drag it to somewhere meaningful on your background image:

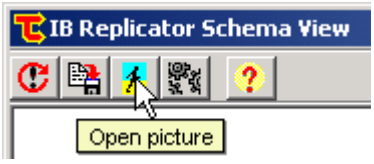
...and then drop it where you want it:

Of course, this would be so much more meaningful to most of us if the map were on a smaller scale! So the tool allows you to use any image you like as your background: perhaps a map of your region, country, plant, server room—whatever works for you!

To Use Your Own Picture as Background

Create or copy the image (.bmp, .jpg, etc.) into a convenient location and scale it to suit.

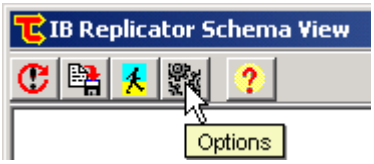
Click the Image button in the toolbar:



This will open a file browser dialog, enabling you to locate and select your image.

Other Layout Options

To open a dialog where you can customise your display, click the Options button:



Remember to Save Your Settings

If you wish to save your layout, simply click on the Save button.

To restore the original (default) layout simply click the Reset button. You can also restore the original layout from the Options dialog, by checking Use Defaults there.

9

Chapter 9 - Advanced Topics

Chapter 9 - Advanced Topics

This chapter examines an assortment of topics that will be of interest to the replication designer when addressing more complex replication requirements. As time goes by, more articles and "How-tos" will be added.

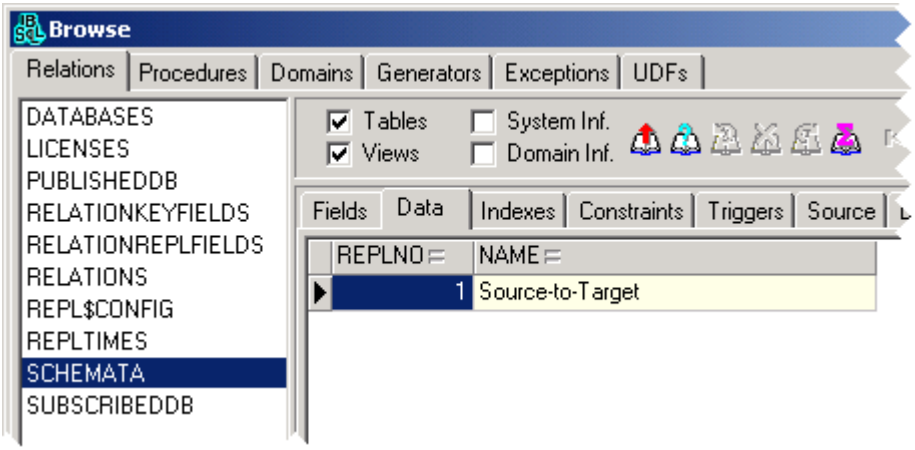
- Schema Numbering 99 explains why the internal numbers assigned to configurations by the Replication Manager might be interesting to the replication designer
- Metadata Changes 102 describes what happens when metadata changes occur in the source database
- Adding Timefields 102 provides a way to generate a script that you can run to add Timefields to all of the tables in a database
- Advanced Mapping Techniques 104 provides a simple example of writing a stored procedure in a target database to accept replication parameters from the source database
- Complex Schemas 106 summarises some more complex replication models and works through an example of n-way replication using the Hub-and-Spoke model

9.1 Schema Numbering

In many replication environments, it is possible to have the same database being referred to as source database in more than one schema. Each schema configuration will cause its own system objects to be created in the source database. Unless precautions are taken, the identifiers of SOs for one schema could conflict with those for another.

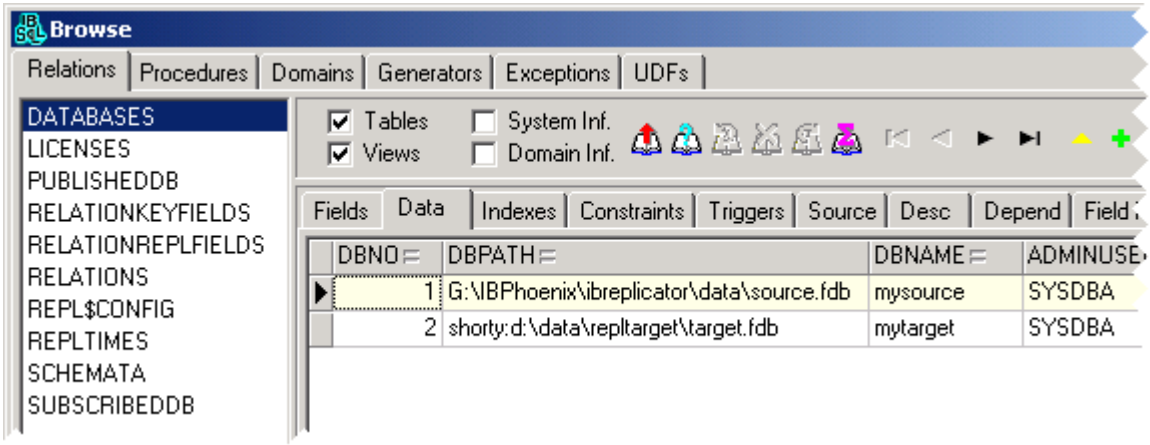
Each schema that is created in a configuration database is referred to by an internal number. This number is used in the naming of triggers, etc., when the System Objects are created. The following screenshots from a graphical DB browser tool illustrate how the numbering is set automatically.

In this shot, we are looking at the configuration database for the replication used in this manual.



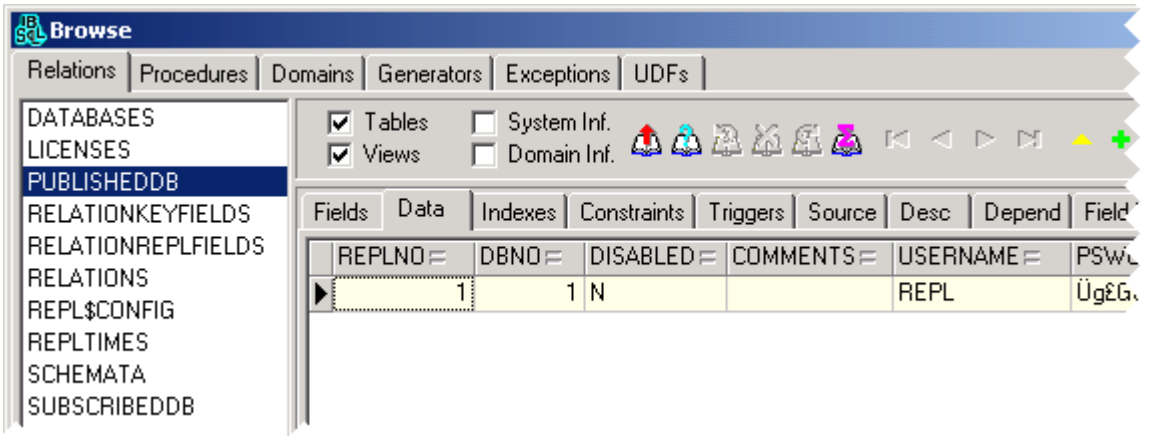
So far, we have only one schema in the schemata table. Its replication number is 1.

Next, we look at the databases defined:



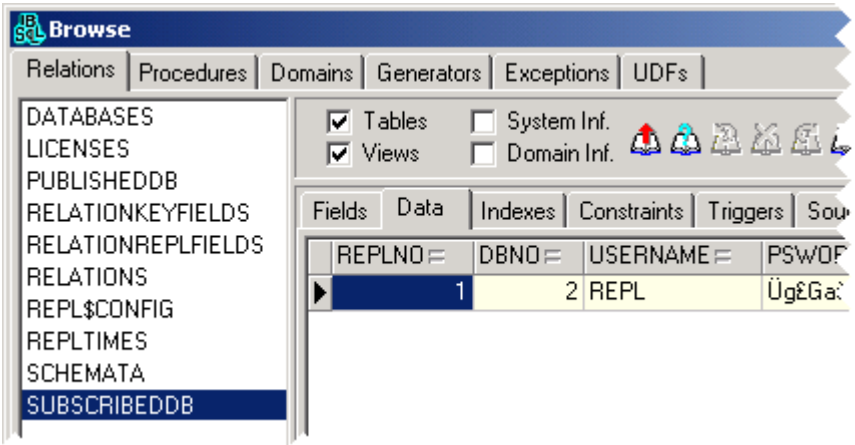
Here we see two databases, in the order that they were defined.

Now, we look at the "published" database—the one that is "publishing" its data to others, i.e. the source:



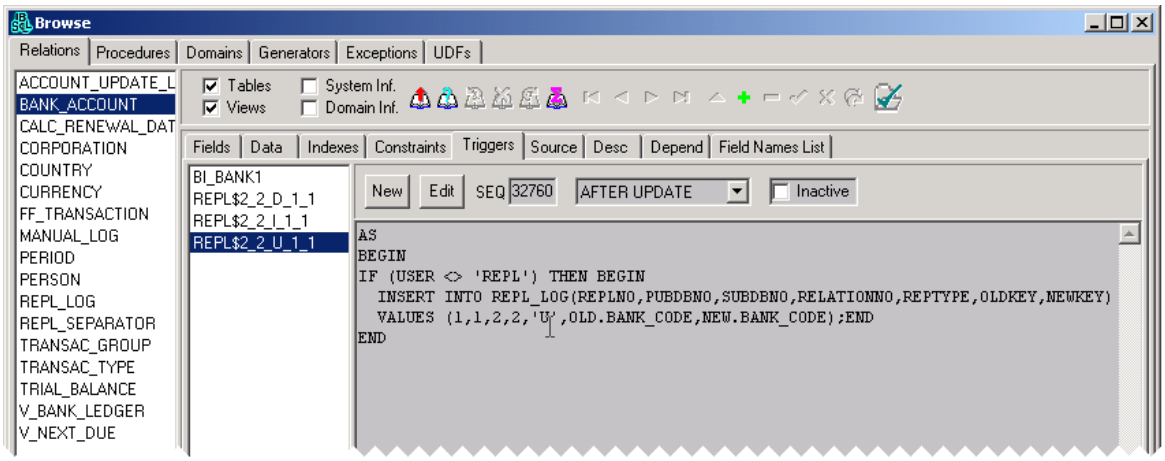
Not surprisingly, we see that our source database is indeed the published one.

And the "subscribed" databases—the targets:



Again, no surprises. Our one and only target database appears.

In the next shot we look at the system objects created in our source database:



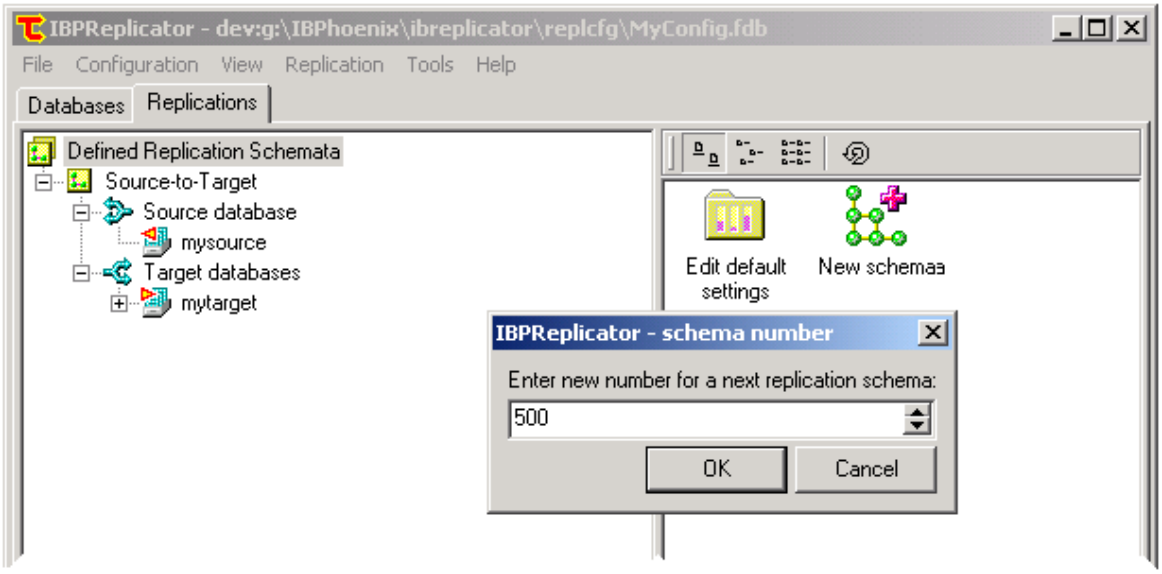
We find three triggers have been created for our replicated tables. The prefix is REPL$2 and the number following it is the replication number of the table—it was the second table we defined in the schema. The two numbers at the ends of their identifiers use the replication number and the database number.

Each configuration database numbers its objects independently. As long as the internal Schema Numbers are unique across all configuration databases, no conflicts can occur.

Setting Your Own Schema Number

When creating a new Configuration Database, you can ensure that the Current Schema Number is set to a number that will prevent two configuration databases using the same numbers. Do this task before you save the new schema record. It is necessary only if one or more source databases are included as sources in more than one configuration database.

Use Configuration | Current Schema No. from the main menu:



For example, ConfigOne.fdb could start at 1, whilst ConfigTwo.fdb could start at 500.

Multiple Schemas within One Configuration Database

Each new Schema created within any configuration database will cause the internal Schema number to be incremented by one—the default Current Schema No. that you see when this dialog pops up is generated internally by a generator called REPLGEN, so there is no risk of conflict with multiple schemas inside the same configuration database. Most configuration databases have fewer than 10 replication schemas defined, so an easy rule would be to start each configuration database with values separated by 100.

> IMPORTANT When you do have multiple configuration databases, you must set one of them to be the default—make it the one to which you want the the replication server connect automatically.

9.2 Metadata Changes

Metadata changes performed after the creation of a replication schema are not propagated between the source and target databases by IBReplicator: you must take care of this by your own devices.

> Replication schemata are not automatically updated to reflect changes in metadata.

Tips for Smooth Changes

- Prepare a DDL script to implement metadata changes in the source database and use the same script, with modifications if necessary, to update the target database[s]. Ad hoc changes performed with interactive tools are prone to human error when duplicated in multiple locations.
- Plan a down-time "window" for the IBReplicator service. Enter the "window" by disabling the IBReplicator service before you run the DDL script on the source. Allow enough time in this window to apply the script to all affected databases, test their effects and modify or replace all affected replication schemata.
- It is strongly recommended that you use a tidy shutdown to remove all users, including other administrative users, from the affected systems while databases and replication configurations are being altered.
- You might wish to archive the replication log file[s] before resuming services.
- Put the database online and restart IBReplicator once you are happy that the revision is complete.

9.3 Adding Timefields

If you plan to use timestamping for as your replication conflict strategy, you will need to add a Timefield column to every source and target table involved in the

© 2007 IBPh en Ed rs

replication and write Before Insert or Update triggers for each table to perform the actual timestamping.

For large databases, this could be a very tedious and error-prone task involving an extended period of downtime for the users at all of the replication sites. However, you can prepare a script ahead of time to speed up the task. You can reduce the effort involved by defining and executing a stored procedure to generate the DDL and export the required statements into an external table.

Generating a Script

Start by defining an external table in the database that the timefield value is going to be applied to:

```
create table ext_data external file 'inject_timestamps.sql' (
  data char(90) character set NONE);
commit;
```

Our sample procedure takes as input the timefield identifier that we have set up for the configuration.

> The generated script depends on all table names having lengths of < 28 characters. If you have any table names that are longer, you will need to edit the script later to replace the trigger names in the generated script.

```
create procedure inject_timestamps (
  timefield_id varchar(31))
as
  declare variable tablename varchar(31);
  declare variable crlf char(2);
  declare variable data varchar(90);
  declare variable date_today varchar(24);
begin
  crlf = '
';
/* literally hit the Enter/Return key between 2 apostrophes. No spaces! */

  date_today = CAST (CAST ('NOW' AS TIMESTAMP) AS VARCHAR(24));
  data = '/* One-time script for applying Timefield metadata to tables';
  insert into ext_data (data) values (:data);

  data = crlf ||' generated by procedure INJECT_TIMESTAMPS ' || date_today || ' */';
  insert into ext_data (data) values (:data);
  data = crlf;
  insert into ext_data (data) values (:data);
  data = crlf ||'SET TERM ^;';
  insert into ext_data (data) values (:data);

  for
    select RDB$RELATION_NAME from RDB$RELATIONS
    where RDB$RELATION_NAME not starting with 'RDB$'
    and RDB$VIEW_SOURCE IS NULL
    and RDB$RELATION_NAME <> 'EXT_DATA'
    into :tablename do
  begin
    data = crlf ||'ALTER TABLE ' || tablename || ' ADD ' || timefield_id || ' TIMESTAMP ^';
    insert into ext_data (data) values (:data);
    data = crlf ||'COMMIT ^';
    insert into ext_data (data) values (:data);
    data = crlf ||'/*    */';
    insert into ext_data (data) values (:data);

    data = crlf ||'CREATE TRIGGER BI5_' || tablename || ' FOR ' || tablename ;
    insert into ext_data (data) values (:data);
    data = crlf ||'ACTIVE BEFORE INSERT OR UPDATE POSITION 5';
    insert into ext_data (data) values (:data);
    data = crlf ||'AS BEGIN';
    insert into ext_data (data) values (:data);
    data = crlf ||'NEW.' || timefield_id || ' = CAST (''NOW'' AS TIMESTAMP);';
    insert into ext_data (data) values (:data);
```

```
    data = crlf ||'END ^';
    insert into ext_data (data) values (:data);
    data = crlf;
    insert into ext_data (data) values (:data);
    data = crlf ||'COMMIT ^';
    insert into ext_data (data) values (:data);
    data = crlf ||'/* ====================== */';
    insert into ext_data (data) values (:data);
  end
  data = crlf ||'SET TERM ;^';
  insert into ext_data (data) values (:data);
  data = crlf ||'/* End script */';
  insert into ext_data (data) values (:data);
  data = crlf;
  insert into ext_data (data) values (:data);
end
```

9.4 Advanced Mapping Techniques

Replicating to Stored Procedures

Sophisticated users can customise IBReplicator to handle specialised replication conditions by defining stored procedures to manipulate the data that the replication will insert, update and delete data in the target database.

Requirements

In designing a Stored Procedure that IBReplicator can use, you need to make provision for everything that will be wanted for the operations.

Input Arguments

Provide one input argument for

- each key column
- each data column that will be replicated
- an OLD value for each key column if an update might be involved
- a final parameter that defines the type of action required: I (insert), U (update), D (delete)

Output Argument

Provide a smallint RETURNS argument that indicates the result of the action. Possible return values are:

0: Success. If the stored procedure handles conflicts itself, e.g., by converting an insert into an update on a key violation, the procedure should also return 0.

1: Unresolvable conflict. The record is moved to MANUAL_LOG.

2: Fatal error. The replication will not proceed.

A Simple Example

The source database contains the following table definition:

```
CREATE TABLE T (
K1 INTEGER NOT NULL,
K2 VARCHAR(10) NOT NULL,
```

© 2007 IBPh en Ed rs

```
K3 DATE NOT NULL,
F1 VARCHAR(20),
F2 DOUBLE PRECISION,
F3 INTEGER,
F4 VARCHAR(100),
F5 NUMERIC(4,1),
F6 NUMERIC(9,2),
F7 NUMERIC(15,2),
PRIMARY KEY (K1,K2,K3)
);
```

A stored procedure can then be defined on a target database with the same structure as the source table as follows:

```
CREATE PROCEDURE REPLICATE_T (
K1_NEW INTEGER,
K2_NEW VARCHAR(10),
K3_NEW DATE,
F1 VARCHAR(20),
F2 DOUBLE PRECISION,
F3 INTEGER,
F4 VARCHAR(100),
F5 NUMERIC(4,1),
F6 NUMERIC(9,2),
F7 NUMERIC(15,2),
K1 INTEGER,
K2 VARCHAR(10),
K3 DATE,
ACTION_TYPE CHAR(1)
) RETURNS (RESULT SMALLINT)
AS
DECLARE VARIABLE ROW_EXISTS SMALLINT;
BEGIN
 RESULT = 0; /*default return value*/
 ROW_EXISTS = 0;  /* initialize */
 IF (EXISTS ( SELECT 1 FROM T
    WHERE K1 = :K1 AND K2 = :K2 AND K3 = :K3))
 THEN
    ROW_EXISTS = 1;

/* Inserts: If the row already exists, then exit with result=1. This
causes the Replication Server to apply conflict rules, which will
probably cause the procedure to be called again, but now with ACTION_TYPE of "U".
An alternative approach would be to simply change the ACTION_TYPE to "U" and proceed
to update the row instead. */

IF (ACTION_TYPE = 'I'
    AND ROW_EXISTS = 1  ))
THEN
   BEGIN
     RESULT = 1;
     EXIT;
   END

/* Updates: If the row does not exist, then exit with result=1. This
causes the Replication Server to apply conflict rules, which will
probably cause the procedure to be called again, but with ACTION_TYPE of "I".
An alternative approach would be to simply change ACTION_TYPE to "I" and proceed
to insert the row instead. */

IF (ACTION_TYPE = 'U' AND ROW_EXISTS = 0) THEN
   BEGIN
    RESULT = 1;
    EXIT;
END

/* Deletes: If the row does not exist then exit with result=1. The
Replication Server will log the error, but otherwise ignore it. */

IF (ACTION_TYPE = 'D' AND ROW_EXISTS = 0) THEN
   BEGIN
    RESULT = 1;
    EXIT;
END

IF (ACTION_TYPE = 'I') THEN
   INSERT INTO T(K1,K2,K3,F1,F2,F3,F4,F5,F6,F7)
   VALUES (:K1_NEW,:K2_NEW,:K3_NEW,:F1,:F2,:F3,:F4,:F5,:F6,:F7);
```

```
IF (ACTION_TYPE = 'U') THEN
  UPDATE T SET
    K1 = :K1_NEW, K2 = K2_NEW,
    K3 = K3_NEW,
    F1 = :F1, F2 = :F2,
    F3 = :F3, F4 = :F4,
    F5 = :F5, F6 = :F6, F7 = :F7
  WHERE K1 = :K1
    AND K2 = :K2
    AND K3 = :K3;

IF (ACTION_TYPE = 'D') THEN
  DELETE FROM T
  WHERE K1 = :K1
  AND K2 = :K2
  AND K3 = :K3;
EXIT;
END
```

9.5 Complex Schemas

Schema scenarios are usually a bit more complex than the simple one used to illustrate the general usage in the earlier topics. In this topic are some tips about approaching the issues when schemas involve n-way replications and heterogeneous scenarios.

Peer to Peer or Bi-Directional Replication

Bi-directional replication is used when source databases need to send changes to target databases and vice-versa. (The target databases become source databases to the original sources, which become targets). Although the concept is straightforward enough, care needs to be taken to prevent changes from “bouncing” backwards and forwards in a continuous loop.

A change made to a source database gets replicated to a target. If the target also acts as a source database to the original source (now a target), this change will get picked up on the target and be replicated back, and so on.

The REPL User

The solution to the looping problem is to have the user REPL defined on the servers.

REPL is a “special” user within the IBReplicator application, as IBReplicator recognises REPL as "myself". Any change made by a REPL user will not get replicated to the target database.

> NOTE To use the REPL user you do need to set the user up in the Firebird or InterBase security database and give the user the appropriate privileges it needs to do its job.

Replicating Subsets of Data

Some scenarios require the rows that are to be replicated from one database to another to be restricted. For example, in a Head Office (London) to Branch Office scenario (Leeds), the branch office needs only to receive replication of data applicable to Leeds—not London specific data, nor data specific to other branches.

© 2007 IBPh en Ed rs

There are two ways to implement this in IBReplicator:

- The simplest is to place a row-level replication condition on each table to exclude all but the applicable rows.

- The other method—usually used for very complex situations—is to replicate all data to a Stored Procedure on the target database and let the logic of the rules implemented in the procedure target database decide whether it wants to receive a row. Refer to the previous topic, Advanced Mapping Techniques 104, for some guidelines for writing such procedures.

Example—Row-level Replication Condition

To illustrate the 'row level replication condition', we will use the following scenario as an example.

The head office is in London, and there are two branch offices in Leeds and Manchester. Sales data needs to be replicated from the branch offices to the head office, and certain other specific data needs to be replicated from the head office to the branch offices.

The Leeds branch office is designated as DeptNo=10, whilst Manchester is Deptno=20.

Manchester (source) -> All Data -> London (target) <- All Data <- Leeds (source)

Manchester (target) <- DeptNo= 20 <- London (source) -> DeptNo=10 -> Leeds (target)

On the Leeds target database, we need to define a row level replication condition for each table, based on the DeptNo. This is done via in the table settings in the Table mapping properties dialog.

```
:DeptNo=10
```

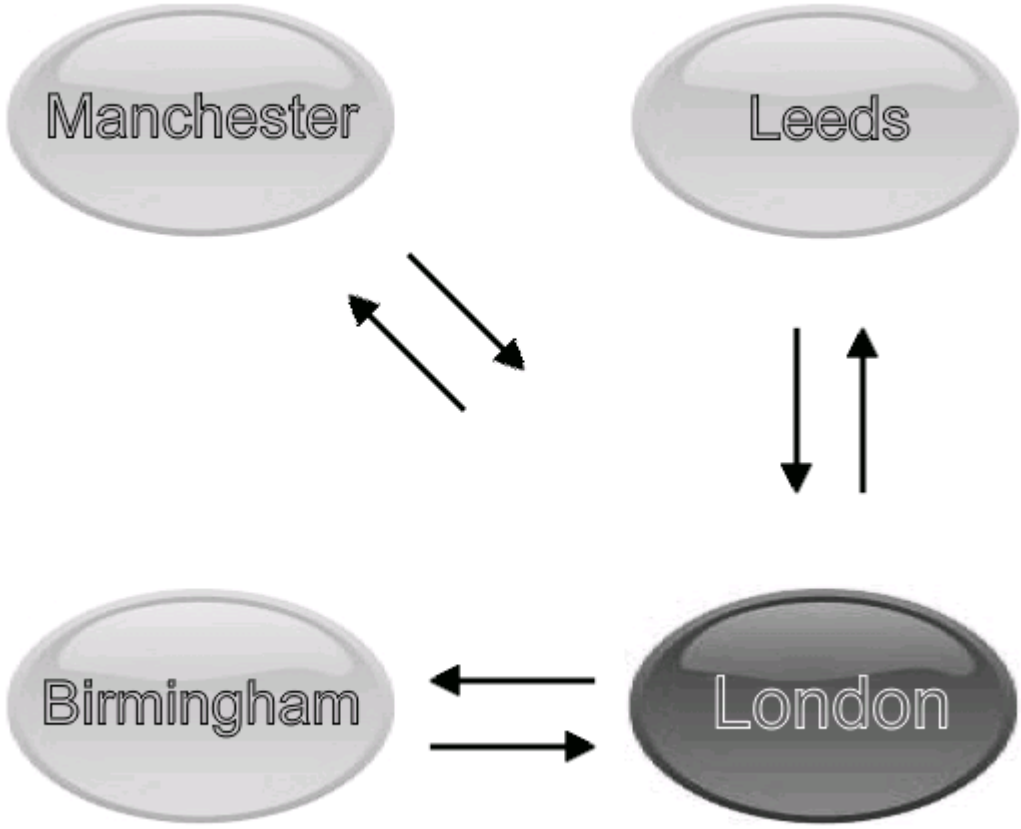
> Note the colon(:) in front of the field name.

`:DeptNo=20` would be applied in the same way to the Manchester target database.

Example—Hub and Spoke Replication

In this scenario a central database controls the replication, fetching and sending data from spoke databases to and from the hub.

© 2007 IBPh en Ed rs



This means that London acts both as a source and as a target for Leeds, Manchester and Birmingham.

In this illustration, London (the central database) fetches changes made by Manchester and forwards those changes onto Leeds and Birmingham.

The same is true for Leeds and Birmingham.

To prevent changes from Manchester/Leeds/Birmingham bouncing back to Manchester/Leeds/Birmingham, respectively, through London, you would normally use the special user REPL in a simple Peer to Peer schema. However that generates a problem, because we do want to forward Manchester's changes to Leeds and Birmingham, Leeds changes to Manchester and Birmingham and Birmingham's changes to Leeds and Manchester.

In this case we cannot use the username REPL, since data would (for example) replicate from Leeds to London, but would not get forwarded to the other branches. On the other hand, we do not want data to be replicated from Leeds to London and then from London back to Leeds.

Our solution is to combine the REPL user, a row level replication condition to exclude the REPL user and some specific user names that we define for doing the replication that REPL is excluded from.

The triggers to handle the Inserts, Updates and Deletes generated in the source

© 2007 IBPh en Ed rs

databases (Manchester, Birmingham and Leeds) are as normal:

```
CREATE TRIGGER REPL$2_1_I_1_1 FOR COUNTRY
ACTIVE AFTER INSERT POSITION 32760 AS
BEGIN
  IF (USER<>'REPL') THEN
  BEGIN
    INSERT INTO REPL_LOG (
      REPLNO, PUBDBNO, SUBSBNO, RELATIONNO,REPTYPE, NEWKEY)
    VALUES(1,1,2,1,'I',NEW.COUNTRY);
  END
END
```

The Source Database Triggers

The triggers created in the source database for London (for each target) look like this:

```
CREATE TRIGGER REPL$1_3_I_3_2 FOR COUNTRY
ACTIVE AFTER INSERT POSITION 32760 AS
BEGIN
  IF (USER<>'REPL') THEN
  BEGIN
    IF (USER<>'REPL_MAN') THEN
    BEGIN
      INSERT INTO REPL_LOG (
        REPLNO, PUBDBNO, SUBSBNO, RELATIONNO,REPTYPE, NEWKEY)
      VALUES(3,2,1,3,'I',NEW.COUNTRY);
    END
  END
END
```

This logic can be tabulated as follows:

Source	Username	Target	Username	Condition
Manchester	REPL	London	REPL_MAN	
Birmingham	REPL	London	REPL_BIR	
Leeds	REPL	London	REPL_LEE	
London	REPL	Manchester	REPL	USER <> 'REPL_MAN'
London	REPL	Birmingham	REPL	USER <> 'REPL_BIR'
London	REPL	Leeds	REPL	USER <> 'LEE'

This ensures that data is replicated from one branch into the hub in London. The hub then replicates this data to the other branches, but not back to the originating branch.

Replication Schema Definition

The defined replication schema for such a scenario would look something like this:

Manchester To London

Source Database in Manchester – UserName REPL

Target Database in London – UserName REPL_MAN

© 2007 IBPh en Ed rs

Birmingham To London

Source Database in Birmingham - UserName REPL

Target Database in London - UserName REPL_BIR

Leeds To London

Source Database in Leeds- UserName REPL

Target Database in London - UserName REPL_LEE

London to the Rest

Source Database in London - UserName REPL

Target Databases

Manchester - UserName REPL, Row Level Replication Condition USER<>' REPL_MAN'

Birmingham- UserName REPL, Row Level Replication Condition USER<>'REPL_BIR'

Leeds - UserName REPL, Row Level Replication Condition USER<>'REPL_LEE'

© 2007 IBPh en Ed rs

10

Version Notes

Version Notes

Replication Server 2.5

IMPORTANT!

The configuration database structure in version 2 releases of this software is not compatible with that of version 1.x releases.

Please take copies of your configuration databases before installing.

The newer Replication Manager will automatically upgrade old configuration databases on opening.

Changes to Note When Upgrading

The Replication scheduler is now built-into the Replication Server and no longer runs as a separate service. Using the Scheduler tool is now the only way to configure and apply a schedule.

You must uninstall a pre-existing scheduler service before upgrading.

Setting a schedule is now the way that IBReplicator replicates automatically. Upgrading an existing configuration to v.2.5 will convert a "regular interval" Replication into a Schedule.

Support for Remote Access Services (RAS) is now enabled. For details, see Registering Databases 37.

A row-level replication condition can now be applied across the entire scope of the target database. This will be especially useful for multi-site scenarios, where records in some or all tables include a site identifier key. See target database Settings 56.

Each target database can now have its own Replication log file. Set this up in the Event Logging tab 57 for the target database.

It is now possible to choose which operations (insert/update/delete) will be performed on each table. Details here 65.

A custom schema order for multi-target replication is now supported—see Adding More Schemata and Ordering Them 49.

Synchronization of all tables in a database can be defined in a customizable order 65.

Separator values for the elements of composite primary keys can now be set individually for each table—see table-level separator character 65.

When performing bulk auto-generation of tables, keys and system objects, you can now optionally select the tables 60 to which it applies.

Other Changes

The configuration database structure has changed to support the new features in 2.5.

© 2007 IBPh en Ed rs

However, the changes should not cause any conflicts in an existing v.2.1.x configuration database.

The global Timefield field name from a target database definition is now used as the default for each replicated table. For this to work, each source and target database table needs a timestamp column with the same fieldname as the one defined in any target database. See Adding Timefields 102.

The Replication Monitor 85 interface has been improved and simplified.

The separate Service Management application has been removed. The installer will automatically set up the service according to the options you select during installation, or as set by command-line switches.

ReplServer for Windows now supports 'c', 'u', 'p' and 'r' switches—see Command-Line Operation 79.

The list of reserved words has been moved into the configuration database and no long uses an external file.

A bug has been fixed so that total statistics are properly reset at start of a replication cycle.

© 2007 IBPh en Ed rs

Index

- A -

Administrative role 33, 36
Architecture of IBReplicator 12
Asynchronous replication 12
Autogenerate mappings 60

- B -

Benefits of IBReplicator 9
Bi-directional 106
Bulk mapping of tables 60

- C -

Cascade 20
Central-to-branch 20
Central-to-standby 20
Choosing,replicated tables 58
Choosing,source database 50
Choosing,target database 54
Columns,identifying 71
Columns,matching 71
Command-line operation,Linux 81
Command-line operation,Windows 79
Command-line switches 79
Configuration database 30
Configuration database,default 42
Configuration,default 42
Conflict resolution 26
Conflict resolution,master-slave 26
Conflict resolution,priority-based 26
Conflict resolution,setting default strategy 43
Conflict resolution,timestamping 26
Conflict resolution,timestamps for 24
Conflict viewer tool 89
Connection,for source database 50
Connection,for target database 54
Context menus 45
Controlling replication 75
Create schema 45
Customising table order 65

- D -

Database features,support for 9
Database,choosing a source 50
Database,choosing a target 54
Database,design issues 24
Databases,registering 12, 33
Default configuration database 42
Default settings 43
Default settings,customising 43
Definitions, ways to work on 46
Definitions,ways to work on 45
Distributed primary keys 24

- E -

Event logging 45
Exclude an operation 65
Executables,Windows 75

- H -

Handling conflicts 26
Heterogeneous replication 26
Hub-and-spoke 20

- I -

IBReplicator,architecture 12
IBReplicator,benefits 9
IBReplicator,security 9
IBReplicator,terminology 12
Installation 29
Installation,Start Menu shortcuts 29
Installation,Windows installer 29
Installer switches 79
Installing licences 91
Intervals,replication 43
IPX/SPX protocol 30

- L -

Licences,installing 91
License Manager tool 91

© 2007 IBPh en Ed rs

Licenses,about 39
Licensing,about 39
Limbo transactions 16
Linux operation 81
Local protocol 30
Log file override 57
Logging 43

- M -

Managing replication 75
Mapping tables, bulk 58
Mapping tables,heterogeneous 58
Mapping tables,removing mappings 58
Mapping,with stored procedures 104
Metadata changes 102
Monitor tool 85

- N -

Named Pipes protocol 30
NetBEUI protocol 30
Network protocol,IPX/SPX 30
Network protocol,local 30
Network protocol,Named Pipes 30
Network protocol,NetBEUI 30
Network protocol,TCP/IP 30
Notify Server utility 88
N-way replication 26

- O -

One-way replication 26
Operations to replicate 65
Operations,Linux 81
Operations,Windows 75
Ordering of replications 49
Override log file per target 57

- P -

Password,for source database 50
Password,for target database 54
Peer-to-peer 20, 106
Planning for replication 20
Primary keys 24
Primary keys,distributed 24
Primary keys,identifying 69
Primary keys,matching 69

- R -

RAS parameters 37
Registered database 50, 54
Registering a database 12
Registering databases 12, 33
Remote Access (RAS) 37
Remote access parameters 33
REPL user 106
Replicant 12
Replicant, definition of 12
Replication 12
Replication Monitor tool 85
Replication order 49
Replication Scheduler tool 86
Replication schema,defining 16
Replication scheme 20
Replication scheme,Cascade 20
Replication scheme,Central-to-branch 20
Replication scheme,Central-to-standby 20
Replication scheme,Hub-and-spoke 20
Replication scheme,Peer-to-peer 20
Replication tree 45
Replication,ad hoc 86
Replication,asynchronous 12
Replication,bi-directional 106
Replication,controlling 75
Replication,executing 16
Replication,heterogeneous 26
Replication,intervals 43
Replication,managed 86
Replication,managing 75
Replication,n-way 26
Replication,one-way 26
Replication,planning for 20
Replication,sequence of events 16
Replication,synchronous 12
Replication,timestamps for 24
Replication,tree 45
Replicator 12
Replicator,definition of 12
ReplServer.exe 75

© 2007 IBPh en Ed rs

Requirements 6
Requirements,database engine 6
Requirements,disk space 6
Requirements,licensing 6
Requirements,minimum database installation 6
Requirements,operating system 6
Role 33
Role, Administrative 36
Role,for source database 50
Role,for target database 54
Row-level condition 64
Row-level conditions 62
Row-level replication condition 64
Row-level replication condition (global) 56

- S -

Scheduler tool 86
Schema View tool 94
Schema, creating 45
Schema, numbering 99
Schema,complex 106
Schema,defining 16
Security 9
Separator character 53, 65
Separator character (global) 53
Separator character (table-level) 65
Sequence of events 16
Source database 12
Source database,choosing 50
Source database,event logging 50
Source database,settings 50
Stored procedures 104
Synchronization 12
Synchronization of tables 65
Synchronization order 65
Synchronous replication 12
System Objects 16
System Objects,creating 72
System Objects,removing 72

- T -

Tables,mapping source and target 58
Tables,synchronization 65
Target database 12
Target database,choosing 54
Target database,General tab 54
Target database,periodic commit 54
Target database,settings 54
TCP/IP protocol 30
Timefields,for timestamping 102
Timefields,generating a script for 102
Timestamps for conflict resolution 24
Tools,Conflict viewer 89
Tools,License Manager 91
Tools,Notify server utility 88
Tools,Replication Monitor 85
Tools,Replication Scheduler 86
Tools,Schema Viewer 94
Tree nodes 45
Two-phase commit 16

- U -

Unique keys 24
Username,for source database 50
Username,for target database 54

- V -

View Conflicts tool 89

- W -

Working on definitions 46

© 2007 IBPh en Ed rs

© 2007 IBPhoenix Editors